The GNU LaTeX Manual

for LaTeX version 2e
Edition 0.3
April 2002

John Arley Burns
Brady Hunsaker

Torsten Martinsen
Piet van Oostrum
Stephen Gilmore
George D. Greenwade

Short Contents

1 Overview 1
2 Using LaTeX 7
3 LaTeX Syntax and Modes 11
4 Typesetting Plain Text 15
5 Logical Document Structure 23
6 Document Layout 25
7 Mathematics 27
8 Computer Programs and Documentation 29
9 Customizing LaTeX 31
10 Packages, Commands, and Internals 33
A Command Reference 35
B Example Documents 85
C Cookbook 89
D Other Resources 91
E Concept Index 93
F Command Index 99

Table of Contents

1 Overview 1

1.1 Why you should use LaTeX 1
1.2 Why you should not use LaTeX 2
1.3 LaTeX versus TEX 3
1.4 LaTeX versus DocBook 4
1.5 LaTeX versus html 4
1.6 LaTeX versus xml 4
1.7 LaTeX versus wysiwyg 4

2 Using LaTeX 7

2.1 Invoking LaTeX 7
2.2 Input and Output Files 7
2.3 Generating Indicies 8
2.4 Making dvi 8
2.5 Making PostScript 8
2.6 Making PDF 9
2.7 Making html 9

3 LaTeX Syntax and Modes 11

3.1 Basic Syntax, Commands, and Environments 11
3.1.1 Comments 11
3.1.2 Command Syntax 11
3.1.3 Environment Syntax 12
3.1.4 Declarations and Scope 13
3.1.5 Skeleton Input File 13
3.2 Modes 14
3.2.1 Paragraph Mode 14
3.2.2 LR Mode 14
3.2.3 Math Mode 14

4 Typesetting Plain Text 15

4.1 Characters to be Careful of 15
4.2 Spacing in the Input 15
4.3 Spacing in the Output 16
4.4 Perfect Punctuation 17
4.4.1 Quotation Marks 17
4.4.2 Dashes 18
4.5 Handling Hyphens 18
4.6 Fonts and Typefaces 19
4.7 Foreign Languages 20
4.7.1 General Foreign Typesetting 21
4.7.2 Typesetting French 22

4.7.3 Typesetting German 22
4.7.4 Typesetting Spanish 22

5 Logical Document Structure 23

5.1 Classes of Documents 23
5.2 Chapters and Sections 23
5.3 Figures 23
5.4 References 23
5.5 Citations 23
5.6 Making a Table of Contents 23
5.7 Appendicies 23
5.8 Indicies 23
5.9 Bibliography 23

6 Document Layout 25

6.1 Lists 25
6.2 Tabular Environment 25
6.3 Quote and Quotation Environments 25
6.4 Verbatim Environment 25
6.5 Verse Environment 25
6.6 Drawing Images 25
6.7 Including Images 25
6.8 Page Layout 25

7 Mathematics 27

7.1 Basic Math in LaTeX 27
7.2 Display Math 27
7.3 Theorems and Equations 27
7.4 Elaborate Math Typesetting 27

8 Computer Programs and Documentation . . . 29

8.1 Basic Source Code Typesetting 29
8.2 Defining Modules 29
8.3 Declaring Functions 29
8.4 Variables and Constants 29
8.5 Literate Programming 29

9 Customizing LaTeX 31

9.1 Documents 31
9.2 Fonts 31
9.3 Custom Lists 31
9.4 Lengths and Measures 31

10 Packages, Commands, and Internals 33
10.1 Loading Packages 33
10.2 Writing Commands 33
10.3 Writing Packages 33
10.4 LaTeX Internals 33
10.5 LaTeX Programs 33

Appendix A Command Reference 35
A.1 Counters 35
A.1.1 \addtocounter 35
A.1.2 \alph 35
A.1.3 \arabic 35
A.1.4 \fnsymbol 35
A.1.5 \newcounter 36
A.1.6 \refstepcounter 36
A.1.7 \roman 36
A.1.8 \stepcounter 36
A.1.9 \setcounter 36
A.1.10 \usecounter 36
A.1.11 \value 36
A.2 Cross References 37
A.2.1 \label 37
A.2.2 \pageref 37
A.2.3 \ref 37
A.3 Definitions 38
A.3.1 \newcommand 38
A.3.2 \newenvironment 38
A.3.3 \newtheorem 38
A.3.4 \newfont 39
A.4 Document Classes 40
A.5 Layout 41
A.5.1 \flushbottom 41
A.5.2 \onecolumn 41
A.5.3 \raggedbottom 41
A.5.4 \twocolumn 41
A.6 Environments 42
A.6.1 array 42
A.6.2 center 42
A.6.2.1 \centering 42
A.6.3 description 43
A.6.4 enumerate 43
A.6.5 eqnarray 43
A.6.6 equation 44
A.6.7 figure 44
A.6.8 flushleft 44
A.6.8.1 \raggedright 44
A.6.9 flushright 45
A.6.9.1 \raggedleft 45

A.6.10 itemize . . . 45
A.6.11 letter . . . 45
A.6.12 list . . . 46
A.6.13 minipage . . . 46
A.6.14 picture . . . 46
A.6.14.1 \circle . . . 47
A.6.14.2 \dashbox . . . 47
A.6.14.3 \frame . . . 48
A.6.14.4 \framebox . . . 48
A.6.14.5 \line . . . 48
A.6.14.6 \linethickness . . . 48
A.6.14.7 \makebox . . . 48
A.6.14.8 \multiput . . . 48
A.6.14.9 \oval . . . 49
A.6.14.10 \put . . . 49
A.6.14.11 \shortstack . . . 49
A.6.14.12 \vector . . . 49
A.6.15 quotation . . . 49
A.6.16 quote . . . 49
A.6.17 tabbing . . . 50
A.6.18 table . . . 51
A.6.19 tabular . . . 51
A.6.19.1 \cline . . . 53
A.6.19.2 \hline . . . 53
A.6.19.3 \multicolumn . . . 53
A.6.19.4 \vline . . . 53
A.6.20 thebibliography . . . 53
A.6.20.1 \bibitem . . . 53
A.6.20.2 \cite . . . 54
A.6.20.3 \nocite . . . 54
A.6.20.4 Using BibTeX . . . 54
A.6.21 theorem . . . 54
A.6.22 titlepage . . . 55
A.6.23 verbatim . . . 55
A.6.23.1 \verb . . . 55
A.6.24 verse . . . 55
A.7 Footnotes . . . 56
A.7.1 \footnote . . . 56
A.7.2 \footnotemark . . . 56
A.7.3 \footnotetext . . . 56
A.8 Lengths . . . 57
A.8.1 \newlength . . . 57
A.8.2 \setlength . . . 57
A.8.3 \addtolength . . . 57
A.8.4 \settodepth . . . 57
A.8.5 \settoheight . . . 57
A.8.6 \settowidth . . . 57
A.8.7 Predefined lengths . . . 57

A.9 Letters 58
A.9.1 \address 58
A.9.2 \cc 58
A.9.3 \closing 58
A.9.4 \encl 59
A.9.5 \location 59
A.9.6 \makelabels 59
A.9.7 \name 59
A.9.8 \opening 59
A.9.9 \ps 59
A.9.10 \signature 59
A.9.11 \startbreaks 59
A.9.12 \stopbreaks 59
A.9.13 \telephone 60
A.10 Line & Page Breaking 61
A.10.1 \\ 61
A.10.2 \- 61
A.10.3 \cleardoublepage 61
A.10.4 \clearpage 61
A.10.5 \enlargethispage 61
A.10.6 \fussy 61
A.10.7 \hyphenation 62
A.10.8 \linebreak 62
A.10.9 \newline 62
A.10.10 \newpage 62
A.10.11 \nolinebreak 62
A.10.12 \nopagebreak 62
A.10.13 \pagebreak 62
A.10.14 \sloppy 63
A.11 Making Paragraphs 64
A.11.1 \indent 64
A.11.2 \noindent 64
A.11.3 \par 64
A.12 Margin Notes 65
A.13 Math Formulae 65
A.13.1 Subscripts & Superscripts 65
A.13.2 Math Symbols 65
A.13.3 Spacing in Math Mode 66
A.13.4 Math Miscellany 66
A.14 Modes 68
A.15 Page Styles 69
A.15.1 \maketitle 69
A.15.2 \author 69
A.15.3 \date 69
A.15.4 \thanks 69
A.15.5 \title 69
A.15.6 \pagenumbering 69
A.15.7 \pagestyle 70

A.15.8 \markboth . . . 70
A.15.9 \markright . . . 70
A.15.10 \thispagestyle . . . 70
A.16 Sectioning . . . 71
A.17 Spaces & Boxes . . . 72
A.17.1 \dotfill . . . 72
A.17.2 \hfill . . . 72
A.17.3 \hrulefill . . . 72
A.17.4 \hspace . . . 72
A.17.5 \addvspace . . . 72
A.17.6 \bigskip . . . 72
A.17.7 \medskip . . . 72
A.17.8 \smallskip . . . 72
A.17.9 \vfill . . . 73
A.17.10 \vspace . . . 73
A.17.11 \fbox . . . 73
A.17.12 \framebox . . . 73
A.17.13 lrbox . . . 73
A.17.14 \makebox . . . 73
A.17.15 \mbox . . . 73
A.17.16 \newsavebox . . . 74
A.17.17 \parbox . . . 74
A.17.18 \raisebox . . . 74
A.17.19 \rule . . . 75
A.17.20 \savebox . . . 75
A.17.21 \sbox . . . 75
A.17.22 \usebox . . . 75
A.18 Special Characters . . . 76
A.19 Splitting the Input . . . 76
A.19.1 \include . . . 76
A.19.2 \includeonly . . . 76
A.19.3 \input . . . 77
A.20 Starting & Ending . . . 78
A.21 Table of Contents . . . 79
A.21.1 \addcontentsline . . . 79
A.21.2 \addtocontents . . . 79
A.22 Terminal Input/Output . . . 80
A.22.1 \typein . . . 80
A.22.2 \typeout . . . 80
A.23 Typefaces . . . 81
A.23.1 \Styles . . . 81
A.23.2 Sizes . . . 82
A.23.3 Low-level font commands . . . 82

Appendix B Example Documents 85

B.1 Skeleton Document 85
B.2 Example Article 85
B.3 Example Report 86
B.4 Example Book 86
B.5 Example Letter 86
B.6 Example Slides 87

Appendix C Cookbook 89

C.1 Nifty Tables with `halign` 89
C.2 Writing a Dictionary for Dummies 89
C.3 Object Oriented Personas 89
C.4 A Neat Program Written in LaTeX 89
C.5 A Literate Program in Perl 89
C.6 Using Make to Build LaTeX 89

Appendix D Other Resources 91

Appendix E Concept Index 93

Appendix F Command Index 99

1 Overview

LaTeX is a freely available, widely-installed, extensible document formatting system. It allows a user to compose a document on an abstract logical level, then have that document converted to several online-formats and/or high-quality printed output. It has been used to create documents varying in size from a small letter to a large textbook. It is available on most operating systems, download-able from the internet, and requires only a simple text browser to write documents in.

LaTeX is *not* a graphical editor, a direct imaging language, or a variant of SGML. It does not allow integrated active spreadsheets, automated database content display, or continuously updated live content. It is not tied to a particular application, operating system, or output format.

1.1 Why you should use LaTeX

So you've gotten this far, and now you're wondering why you should use LaTeX, Maybe you already know why, in which case you can skip this section. But it may prove worthwhile to read on, because LaTeX is often used for reasons you may not have thought of before. You may even find new ways in which you can use LaTeX.

In my experience, most people use LaTeX because they're told to use LaTeX in their school or business. Usually, the people who worked on the project before you used LaTeX, and now you need to learn it. This is true of both technical writers and programmers, since LaTeX documents often intersect these disciplines.

There has to be a reason LaTeX got used in the first place, however. And maybe you are evaluating whether to use LaTeX or not. Mainly people began using LaTeX because it was freely available, standard, cross-platform, simple to use, powerful, abstract, extensible, proven, and produced high-quality output. Before LaTeX many other systems achieved some of these goals, but LaTeX, in my opinion, was the first to achieve all of these reasonably well.

Free The LaTeX system is freely available. With LaTeX, you don't need to pay a lot of money just to get started, and you don't need to keep cash flowing every time a new release comes out. The LaTeX system is freely available across the internet. In addition, the source code is openly distributed, so you won't be locked into the way one company or group decides to do things in the future. And you are free to modify LaTeX to your liking.

Standard The LaTeX system is standard. It exists in many different installations, operating systems, and countries all over the world, with the same document format. A LaTeX document can be run on any LaTeX system, with the same output and behavior. And the LaTeX standard is supported by numerous organizations and individuals, a truly cooperative effort with no dominating monopoly.

Cross-Platform
The LaTeX system is cross-platform. With LaTeX, you don't need to get locked in to a proprietary product that may work on only some of your platforms. You don't need to have different document standards for different systems, along with the added cost of training and support for these systems. With LaTeX, you can have one document format for all your platforms.

Simple The LaTeX system is simple to use. Once installed, the only input needed is an ASCII text file. Any number of editors can produce this, and some editors such as Emacs have special methods of making LaTeX easy to use. Graphics may be created by a tool such as Gimp and included via the Encapsulated PostScript standard, which can be output with readily available tools. Basic LaTeX documents can be produced by a user after only a short period of training.

Powerful The LaTeX system is powerful: it can produce all kinds of documents. It can produce a letter to a politician, an article for a mathematics journal, a report on your latest business proposal, or a full-length book with a myriad of features, and even slides for a presentation. Documents can include tables, glossaries, multiple languages, color graphics, automatically generated table of contents and indices, and more.

Abstract The LaTeX system is abstract: it allows one to compose documents on a high level. In many systems the user can get bogged down early on with fonts, formatting, style, how the document prints, and other non-semantic items. With LaTeX, the user can focus on writing the document first, getting all the information in, and only later be concerned with specific formatting. In fact, LaTeX's default formatting is so good that it is often unnecessary to concern oneself with particular display questions. This abstract nature of LaTeX translates into more time spent writing quality documents, and less time fudging the output.

Extensible The LaTeX system is extensible. This means you can extend it with new commands, styles, even languages and output formats, all while being compatible with the LaTeX standard. If you find LaTeX doesn't do something you like, or you want to do something more efficiently, there is no barrier to this as in most other document systems. LaTeX itself is Turing-Complete, so you are really quite unlimited in how you can extend LaTeX.

Proven The LaTeX system is proven. Since the mid-eighties LaTeX has been used in countless publications, articles, technical documents, and textbooks. It is unusually solid and bug-free. In addition, it is constantly being upgraded, and the quality standards for new releases are quite high. Throughout its history, LaTeX has shown itself to be a proven and reliable product.

Quality The LaTeX system produces high quality output. All the previously mentioned qualities of LaTeX become quite meaningless if high quality output cannot be generated, but this is where LaTeX really shows superiority. Since LaTeX sits upon the TeX system, it uses the TeX algorithms to produce ideal printed output. These algorithms were developed after a long and thorough study of typesetting by Donald E. Knuth, one of the most famous computer scientists and writer of the TeX system. The LaTeX system was built upon TeX by Leslie Lamport, to make TeX more powerful and easier to use. But in addition to this unsurpassed typeset output, which can be viewed from a printer, DVI previewer, or in industry-standard PostScript and PDF, LaTeX can also produce plain ASCII and HTML.

1.2 Why you should not use LaTeX

There are several good reasons not to use LaTeX. Though in general LaTeX is one of the best, if not the best documentation system, there are weaknesses that may cause a user to avoid it. The reasons for not using LaTeX are given here.

Learning You may not have the time to learn LaTeX. Though it is by no means a complex or obscure language, it does take some effort to learn the basic commands and formatting techniques. This may be particularly difficult if you are used to a WYSIWYG editor, and have never used a formatting language.

Desire You may not have the desire to leave your current document formatting system, particularly if it is WYSIWYG. It may not be worth the advantages of LaTeX to switch.

System You may be on a computer system where LaTeX is not available, and where a system administrator is unwilling to install it. You can install LaTeX yourself, but this can be difficult.

Inter-operable
There may be an existing document standard with those you work with, and you have to use their document standard. If this standard is not LaTeX, you will be hard-pressed to use it.

Audience The language you develop your document in is often affected by the target audience. Texinfo documents, for instance, can be converted to a wide variety of formats much more easily and reliably and completely than can a LaTeX document.

1.3 LaTeX versus TeX

It is important to distinguish LaTeX and TeX; the two are highly related, but are nonetheless distinct. You can use either one to produce documents, though they are different styles of formatting.

TeX was created in the eighties by Donald Knuth to create a superior computer typesetting system. The focus was on producing very high quality book output, and was originally meant to typeset Knuth's *The Art of Computer Programming*. It has been quite successful, and used widely for scientific books, articles, and in literate programming. There are many ways to use TeX, from bare low-level commands, to general macro packages such as Plain TeX, to specialized packages such as AMS TeX.

LaTeX is simply another macro package for TeX, though admittedly the most general, powerful, and widely used. LaTeX commands tend to be higher-level than TeX commands. They include such useful default document types as book, article, report, and letter, and can even be used for writing slide presentations. Importantly, LaTeX is much more general than TeX, so that LaTeX can be more easily converted to alternative formats such as HTML. In addition, LaTeX can still be processed through TeX, giving high-quality printed output. Because of these reasons, LaTeX is now the most widely used TeX macro package, and used much more frequently than bare TeX.

1.4 LaTeX versus DocBook

this section should be revised by someone's who's used DocBook

1.5 LaTeX versus HTML

LaTeX and HTML are very different in both purpose and extent. They both can be the best solution for a given job, although LaTeX can in fact (usually) be converted to HTML; the reverse is not true.

The purpose of HTML is related to its history. It began in 1991 as an SGML DTD for internet-based hypertext via the accompanying HTTP protocol. The goal was maximum portability across systems, small size of documents, online (as opposed to printed) availability, and focus on content rather then presentation. Though HTML has changed since then, almost into a 'poor man's PDF', this basic purpose of simplicity and portability has remained prevelant in the underlying standards.

The extent of HTML was also affected by this purpose. Documents were expected to be of small size, with little in the way of support for large or complex documents. Basic typography such as tables, font selection, page layout, and printed matter were not supported, and even now only have limited and (at least for publish-quality books) unsatisfactory support. Note that this is not a weakness of HTML per se; it is more a case of writers attempting to extend the format far beyond what it is designed for.

If the purpose of your document coincides with the purpose outlined above, and the extend of your document matches that above, then HTML is probably a good choice for your document. However, if they do not coincide, particulary if you want a document suitable for publishing, then LaTeX is probably a better choice.

An added advantage of LaTeX is that it can be converted into HTML via the program '`latex2html`'. Although this doesn't always work, particularly for large or complex documents, it nonetheless is usually sufficient. So, if you need HTML for a web page of your document you can still use LaTeX as the source language (although honestly Texinfo will work better for HTML conversion).

1.6 LaTeX versus XML

this section should be revised by someone who knows about XML document composition

1.7 LaTeX versus WYSIWYG

A common complaint of those coming across LaTeX for the first time is that it's not WYSIWYG. Though there are some philosophical issues here that won't be addressed, there are great advantages to the fact that LaTeX is *not* WYSIWYG. However, we will try to fairly state the differences here.

It is important to note that LaTeX can be used with a WYSIWYG interface. In fact, such a product exists, called *LyX*, but as it is not truly free software it is not recommended. For more information see `http://www.lyx.org`. It will not be further discussed in this document.

Most users have encountered a WYSIWYG editor at some point. Common editors include MSWord, WordPerfect, and StarOffice to name a few. The basic principle of these editors is that what is displayed onscreen in the GUI is approximately what you will see on processed output. Note that in the case of processing as PostScript or HTML you will in fact usually see something slightly different from what is onscreen. Even with a PostScript display engine the different color dynamics, exact page size, and resolution of a screen versus printed output will often show differences, to the suprise and confusion of many users. So, the weakness of WYSIWYG editors is that what you see is not always what you get.

Another common experience of users with a WYSIWYG editor is difficulty creating and maintaining the logical structure of the document. Since the focus in such editors is on final output, there is little regard given to the logical structure of the document. Such divisions as chapter, section, paragraph are ad-hoc and visually oriented; it is difficult to maintain, and changing styles often means going through the entire document manually. Such useful constructions as an automated index and table of contents are not well supported and sometimes impossible. This is a frustrating experience of most WYSIWYG users, but often they don't know of another option.

Another option is LaTeX. It provides logical structure with many automated features, such as index and table of contents. Page layout, including optimal paragraph filling, is accomplished automatically and with far better results than most other solutions. In addition, any desired feature can be programmed in through LaTeX (and TeX) primitives. Exact control, something never truly achieved in WYSIWYG, is possible if desired (but the automatic formatting is so good this is hardly necessary).

There are still reasons you may prefer WYSIWYG. If your current solution is easy to use and you never use it very often, it may be the best thing for you. You may be forced to use an editor by your existing organization (sadly, this is often true); you may have no choice. But if you edit documents often, especially large documents or those with complex or mathematical layout, you should take a look at LaTeX.

2 Using LaTeX

Unlike many document formats, LaTeX is platform and application independant. That means there are different ways you can prepare the LaTeX document for viewing and/or printing, in a variety of output formats. Such standard formats as PDF, PostScript, and HTML are all supported.

So first decide if you want to produce just one or many formats. Often just one format will do, especially for personal use. Maybe you have a PDF viewer (such as Adobe Acrobat or xpdf) that suits your needs entirely, so you just want to generate PDF. Or maybe you have a broader project where you want the material available on the web in as many formats as possible. If producing just one format, pick it and go to the section indicated below. If you want many formats, it is best to setup your own makefile for generating documents.

PDF	See Section 2.6 [Making PDF], page 9.
DVI	See Section 2.4 [Making DVI], page 8.
PostScript	See Section 2.5 [Making PostScript], page 8.
HTML	See Section 2.7 [Making HTML], page 9.

2.1 Invoking LaTeX

2.2 Input and Output Files

The LaTeX command typesets a file of text using the TeX program and the LaTeX macro package for TeX. To be more specific, it processes an input file containing the text of a document with interspersed commands that describe how the text should be formatted.

There is one main input file to LaTeX. This input file may also use other files, either directly or indirectly, for such extensions as multi-file documents, included images, and generated lists such as table of contents or indicies. The following are common input files:

1. The main source file, usually with a '`.tex`' or '`.ltx`' extension. Any included LaTeX files will have the same extension.
2. Image files are typically encapsulated postscript, with an '`.eps`' extension. With some processors, other image formats can be used, such as JPEG (extension '`.jpeg`' or '`.jpg`').
3. A table of contents, a list of tables, and a list of figures are automatically generated by LaTeX. These have extensions '`.toc`', '`.lot`', '`lof`', respectively. Note that they will only be generated when LaTeX commands exist to create their entries.
4. An auxiliary file with an '`.aux`' extension. This is used by LaTeX itself, for things such as sectioning.
5. An index file with an '`.idx`' extension. This is actually not generated by LaTeX itself but by the external command `makeindex`.

Even though many files are input by LaTeX, the final output is only one file for display or printing (a DVI file, or PDF when using `pdflatex`). Even if multiple input files were used (such as images or multiple source files), this is all processed into one output file. Some files are not directly output, but are generated from another output file (such as a postscript file). The only other direct output file is a log file.

1. A "Device Independent", or '`.dvi`' file. This contains commands that can be translated into commands for a variety of output devices. You can view the output of LaTeX by using a program such as `xdvi`, which actually uses the '`.dvi`' file.
2. A "Portable Document Format", or '`.pdf`' file. This is a more widely supported cross-platform document format than '`.dvi`', but is not as frequently used to generate postscript (often for printing) as the '`.dvi`' file.
3. A postscript file (extension '`.ps`'). This is not directly generated by LaTeX, but rather is converted from another output file (typically converted from the '`.dvi`' file using the command `dvips`.
4. A "transcript" or '`.log`' file that contains summary information and diagnostic messages for any errors discovered in the input file.

2.3 Generating Indicies

Generating indicies is rather simple. You just need to put index references in your document and then use `makeindex` to process them.

Putting index references into your document is easy. Just

2.4 Making DVI

Creating DVI documents is the default behaviour of most LaTeX installations. Just run the `latex` command on the source document, and after a bunch of document information you'll have a new file with a *.dvi* extension. For example, if your document is called '`foo.tex`':

```
latex foo.tex
```

This command will produce a file '`foo.dvi`'. You can then display this file using a DVI viewer such as `xdvi`, or convert it to postscript for viewing or printing (see Section 2.5 [Making PostScript], page 8).

2.5 Making PostScript

Once you have a DVI file, you can proceed to make a postscript file from it (if you don't have a DVI file yet, see Section 2.4 [Making DVI], page 8). Making a postscript file is simple, but be careful that you are using the right options to the DVI to postscript command, `dvips`[1]; a good safety measure is to always use the `-o` option unless you want to risk immediate printout. For example, if you previously created a file '`foo.dvi`', you can convert it to postscript using:

```
dvips -o foo.ps foo.dvi
```

There are many options for `dvips` that allow a wide range of sizing, printing, and output options. Consult a manpage for more info.

You can further process this postscript file into a PDF via the `ps2pdf` command, but it isn't recommended. Instead, use a direct path via `pdflatex` (see Section 2.6 [Making PDF], page 9).

[1] At one point one of the authors switched to a different CS cluster where `dvips` had a different set of default options than he was used to. You can imagine the suprise he felt when what was supposed to be a postscript file generation turned into an automated 200-plus page printout! Needless to say, fellow users were not amused at my quickly amassing paper debacle, and the job had to be cancelled (with embarassment) manually. Caveat agitator!

2.6 Making PDF

The Portable Docmuent Format, commonly known as PDF, is about the closest thing to a searchable, structured and platform-independant document format currently existing in the computer world. High quality readers such as Adobe Acrobat and '`xpdf`' can be downloaded for almost any platform. The PDF format is very faithful across platforms for viewing and printing (in fact in many states legal documents can be received in one of two formats: paper and PDF). As an added plus or minus (depending on your view), the PDF format has no programming constructs such as PostScript, so you don't have to worry about your document displaying in exponential time (unless of course you are low on memory, or using a buggy PDF reader).

Enough raving. There are good reasons not to use PDF. It tends to be larger than some other formats (like HTML). For certain tasks, such as printing, other formats such as PostScript are more suitable. So this is your last chance to bail out before we tell you how to make PDF from LaTeX.

First, you need to make sure that you have '`pdflatex`' installed. On unix, this can be achieved easily with the command:

```
type pdflatex
```

If this doesn't work, you don't have '`pdflatex`' installed or at least not in your path. Contact your system administrator to get it installed; it's part of the standard TeX distributions so there's really no excuse not to. If you're using a GNU operating system such as Debian Linux or Hurd, you can get a new package list and update your TeX package; '`pdflatex`' should be there.

Second, assuming you have '`pdflatex`' available, you should now go to the directory tha has your LaTeX file in it. If you've never done this before, you should use the sample LaTeX article (see Section B.2 [Example Article], page 85). Let's say your LaTeX file is named '`foo.tex`', then to compile your document to PDF you would type:

```
pdflatex foo.tex
```

Type it again, to put in the references. This is a two-pass process: the first pass to do preliminary formatting (table of contents, references, etc) and the second pass to incorporate all the information from the first pass.

There should be a new file now in the current directory called '`foo.pdf`'. You should now be able to view this with your PDF viewer.

2.7 Making HTML

3 LaTeX Syntax and Modes

Before we get to the details of creating a LaTeX input file, it's worth learning a little about the syntax LaTeX expects. This makes it easier to talk about using LaTeX, and it may help you understand why certain things are done the way they are.

3.1 Basic Syntax, Commands, and Environments

The syntax LaTeX expects is fairly simple. Most of what appears in an input file are letters or characters that are meant to be typeset. Words are separated by *whitespace* (spaces, tabs, and newlines) or by spacing commands (add a reference).

The following ten characters, however, are used by LaTeXfor formatting and are known as *special characters*:

```
{ } \ % ~ # & $ ^ _
```

To use these symbols in your output, see Section 4.1 [Characters to be Careful of], page 15.

The rest of this section covers the most important concepts of LaTeX syntax.

3.1.1 Comments

Comments are included in the input file by using the character '`%`'. Once a '`%`' is encountered, everything on the rest of that line of the input file is ignored by LaTeX. In fact, the newline at the end of the line is even removed, so the line break doesn't count as whitespace like it usually would. This allows for the possibility of splitting words over multiple lines, though commands may not be split in this way.

Note that there are a few situations in which a '`%`' doesn't start a comment, such as in the command '`\%`', which is used to actually print a '`%`'.

3.1.2 Command Syntax

Commands are the basic building block of LaTeX formatting. Commands consist of the character '`\`' followed either by a series of letters or by a single non-letter. Examples of the former type of command are `\section` and `\LaTeX`. Examples of the latter type are the commands `\\`, `\&`, and `\^`. Commands are case sensitive, so that `\FooBar`, `\foobar`, and `\FOOBAR` are all different.

Commands may take any number of required arguments, which follow the command name and are each enclosed in braces ('`{`' and '`}`'). A command may also have optional arguments, which often come before any required arguments and are enclosed in brackets ('`[`' and '`]`'). A very few commands have arguments that are enclosed in parentheses ('`(`' and '`)`').

For example, the command `\begin{document}` starts the body of every LaTeX document. A more complicated example is `\newcommand{\HiThere}[1]{Hi there, #1!}`.

In general, LaTeX expects a command whenever it encounters a '`\`'. If the character immediately following the '`\`' is a non-letter, then LaTeX interprets the command name as containing the '`\`' and that single character. If the '`\`' is followed by a letter, then LaTeX interprets every letter until the first non-letter as part of the command name.

For example, in the text '`Ol\'e!`', the command `\'` (which produces an acute accent) is easy to identify since it is a non-letter. Consider the following text with lettered commands:

```
I would like a foo\bar.  Foo\bars are cool.
```

The two commands are `\bar`, which is terminated by a period, and `\bars` which is terminated by a space.

An important detail is that since space may be used to end the command, LATEX ignores any space after a command (it assumes you were just using that space to end the command). So in the foobar example above, there would be no space between the command `\bars` and the following word '`are`'.

Since we actually meant to have space, we can use the command '`\ `' (backslash followed by a space) to explicitly tell LATEX to insert an interword space:

```
I would like a foo\bar.  Foo\bars\ are cool.
```

What if you really wanted to use the command `\bar` again instead of `\bars` in the above example? Perhaps the two most common solutions are as follows:

```
Foo\bar s are cool.
Foo{\bar}s are cool.
```

These two lines would produce exactly the same output. In the first line, a space is used to end the command `\bar`, but LATEX doesn't insert any space there, so the '`s`' is still part of the same word. In the second example, braces are used to make it clear what the command is and no space should be present before the '`s`'. The braces will not appear in the output. For another use of braces, see Section 3.1.4 [Declarations and Scope], page 13.

Note that if a command has arguments, the arguments clearly indicate the end of the command and there is no problem. In this case, whitespace that follows the arguments does count as space.

3.1.3 Environment Syntax

Environments are used to indicate sections of the document that are a logical unit, often a unit that should be typeset differently from the surrounding document. For example, the '`quotation`' environment is used for a long quotation, which would typically be indented more than the surrounding text. Other example environments include '`itemize`' and '`enumerate`' for making lists, '`verbatim`' to simulate typed text, and '`equation`' for displaying equations. These and many other environments will be discussed throughout this document.

An environment is begun with the command `\begin{environment-name}` and is ended with the command `\end{environment-name}`, where '`environment-name`' is replaced with the name of the environment. In a few cases, the environment also takes one or more arguments, which are appended after `\begin{environment-name}`.

There are several effects of using an environment. It is often the case that all of the content in an environment is typeset differently, such as with a different font or different margins. Whether or not this is true, certain commands may only make sense (or be available) in certain environments, while other commands may mean something different inside a certain environment. An example of this is the set of commands used in the '`tabbing`' environment (??? add a reference).

Environments may be nested one inside another. For example, the '`document`' environment contains the body of your document and all other environments must be placed within it. Environments should not overlap, however, unless one is nested inside the other (??? check on this).

3.1.4 Declarations and Scope

Many commands only affect their arguments. For example, bold face text can be made with the command `\textbf{}`. The contents of the argument are typeset in bold.

```
Whatever you do, \textbf{don't push the red button}.
Remember that our next rehearsal is on \textbf{Monday, February 31st}.
```

LaTeX also contains commands that have no arguments but affect all the text that comes after them. These commands are called *declarations*. An example is the declaration `\bfseries`, which is another way to produce bold text. Once you include `\bfseries`, text will be typeset in bold forever.

OK, not really forever. The *scope* of the declaration refers to the section of input for which that declaration is in effect. This starts at the point of the declaration, of course, and ends in one of two ways. The first (and simpler) way for the scope to end is for an overriding declaration to occur. For example, a `\bfseries` declaration's scope will end if a `\mdseries` declaration occurs, since this returns the text to the default medium weight.

The other way for the scope of a declaration to end is for a grouping structure that contains the declaration to close. For example, if the declaration occurred inside a '`quotation`' environment (or any other environment), then the end of the environment will be the end of the declaration's scope (if it hasn't ended already).

In addition to environments, the special characters '`{`' and '`}`' can be used to enclose groups of text for scope purposes. They serve this purpose when they enclose command arguments, but they can also be used independently. Each '`{`' must have a matching '`}`' in the input file, and neither one will appear in the output. If a '`{`' occurred before the declaration, then the matching '`}`' will end the scope of the declaration if it hasn't ended already.

There are a few other grouping structures that can end the scope of a declaration. For example, each cell of a tabular environment is in its own scope (??? add a reference).

Note that grouping structures that start *after* the declaration will lie entirely in the scope of the declaration, and the end of that group will not end the declaration's scope. Thus, the only grouping structure that is relevant is the one that *began* most recently before the declaration occurred. Consider the following contrived example:

```
What in the {\itshape world} were you thinking?
\begin{itshape}
I can't \bfseries believe \emph{it}.
\end{itshape}
```

The `\itshape` declaration only affects the word '`world`', since braces are explicitly used to limit its scope. The scope of the `\bfseries` declaration, on the other hand, ends with the '`itshape`' environment, since that is the group that began most recently before the declaration. The command `\emph{}` lies entirely within the scope of the `\bfseries` declaration.

3.1.5 Skeleton Input File

Every LaTeX document must contain a certain basic structure. This structure is shown in the appendix in Section B.1 [Skeleton Document], page 85. There are comments added to describe the various parts of the document. These comments, of course, are not necessary in any documents.

3.2 Modes

LaTeX automatically handles spacing and line breaking in the output. How it performs these tasks as well as some other details depend on which of three modes it is in: paragraph mode, LR mode, or math mode.

3.2.1 Paragraph Mode

This is LaTeX's usual mode. In paragraph mode, LaTeX puts normal spacing between words and breaks lines based on the current margins. In most documents, the bulk of the input file is processed in paragraph mode.

3.2.2 LR Mode

LR mode is short for left-right mode. In this mode, LaTeX puts normal spacing between words but does not break lines. All text in a section processed in LR mode will be typeset on a single line. It's the responsibility of the user to make sure that the content will fit on a single line (as usual, LaTeX will issue a warning if the line is too big).

Examples where LR mode is used include ???

3.2.3 Math Mode

Not surprisingly, math mode is intended for typesetting mathematics. In this mode, LaTeX does not perform the usual spacing between words. In fact, all spacing in the input is ignored, and spacing is based heavily on the specific characters. Most letters and characters are treated as variables, and putting them next to one another is interpreted as multiplication. The spacing involved in multiplying variables is very different from the spacing for words. Operators such as '`+`', however, will have more space around them.

The change in interpretation is also typically reflected by a change in the typeface. Most often variables appear in a slanted font (similar to italics). There are also a large number of symbols available only in math mode.

Line breaking in math mode depends on the specific situation. Usually LaTeX does not perform any line breaking in math mode, but a notable exception is the '`math`' environment, which is commonly used in the middle of normal text.

4 Typesetting Plain Text

Most normal text can be entered just as you would in any text file. Take a look at the example documents in Appendix B [Example Documents], page 85.

There are just a few points to remember:

- Certain special characters are reserved.
- Paragraphs are separated by blank lines. Other spacing in the input file doesn't matter.
- LaTeX sometimes needs a little help to determine correct spacing in the output.
- Some punctuation, such as quotation marks and dashes, are typed differently than you may expect.

These points and a few others will be covered in more detail in this chapter.

4.1 Characters to be Careful of

The following ten characters have special meanings in the input file and are typically referred to as *special characters*:

```
{ } \ % ~ # & $ ^ _
```

For example, '`{`' and '`}`' are used to enclose blocks, while '`\`' is used to begin a command.

If you actually want these symbols to appear in your output, you need to do something different. For the seven symbols '`{ } % # & $ _`' you can simply precede the symbol with a '`\`'. For example:

```
Dewey, Cheatam \& Howe
I live in apartment \# 42.
It cost \$5.
```

The three symbols '`\ ^ ~`' are trickier. You can't use `\\`, since this is used to indicate a new line. LaTeX provides the command '`\backslash`' to produce a backslash, but it is only available in math mode. Therefore, to use a backslash in normal text, use the code `$\backslash$`.

You also can't use `\^` or `\~` since these are commands that tell LaTeX to put accents on the next character (can you guess what the two accents look like?). There are a couple of ways to get around this. Perhaps the simplest is to put the accents over a space, so all you see are the symbols themselves. This can be done by giving the accent commands empty arguments: `\^{}` and `\~{}`.

Note that there are a few environments, such as '`verbatim`', in which special characters are actually printed. In this case it's not necessary (or correct!) to use the alternate commands given here.

4.2 Spacing in the Input

As mentioned, one or more consecutive blank lines in your input tells LaTeX to start a new paragraph. Other than that, you don't have to worry about breaking lines, indentation, or spacing in the input file, since LaTeX ignores it and formats the output automatically. In later chapters we'll also see how you can take control of these formatting issues if you want. Actually there are a few instances in which LaTeX needs a little help, which we'll mention in the next section. The vast majority of the time, however, LaTeX automatically does a superb job. This is (part of) what TeX and LaTeX were designed for!

Specifically, line breaks are ignored except in the case of a blank line, and other cases of consecutive whitespace characters (such as spaces, tabs, and newlines) are treated as a single space.

For example, the following two input fragments would produce the same output:

First example:

```
This text produces a single paragraph.  It doesn't matter how the
lines are broken or how spacing is done.  Do you really want to worry
about those things?

The blank line(s) before this line starts a second paragraph.
```

Second example:

```
This     text produces a
         single
paragraph.            It
doesn't  matter    how     the  lines are
broken
or
how spacing is done.  Do you   really want to       worry
about those things?


The blank line(s)     before this line
starts a        second paragraph.
```

4.3 Spacing in the Output

LaTeX handles normal spacing and line breaking automatically, though it occasionally needs a little help. Later on we'll see how you can take control of spacing, whether to insert extra horizontal or vertical space or perhaps to redefine LaTeX's default behavior. In this section we concentrate on a few common cases where LaTeX may get confused in its default behavior.

Almost all of the confusion is in determining when a sentence has ended. This is important since printed text often (but not always) has a little more space after a sentence than between two words in the same sentence. Remember that LaTeX ignores the spacing in your input file, so it has to figure out from the text when a sentence has ended.

The simple rule that LaTeX uses is that a sentence is ended when a period followed by whitespace occurs, unless the period follows a single capital letter. This is because a single capital letter followed by a period is more likely to be an initial than the end of a sentence. Actually, extra space also goes after exclamation points, question marks, and colons, and all of the details we are about to discuss apply to them as well.

The basic rule works most of the time, but is subject to two types of errors as shown in the following examples.

```
Where is Mr. Rogers' neighborhood?
Free software operating systems (e.g. GNU/Linux) are great!
```

```
I like the poetry of e. e. cummings.

You know better than I.
My middle initial is K.
The correct answer to the sample question is C.
```

Each of the first three examples has one or more periods that do not end the sentence, though LaTeX will think they do. Each of the latter three examples ends a sentence with a single capital letter, but LaTeX will assume the sentence continues.

The solution to the first problem is to follow the period with the command `\ ` (backslash followed by a space). This command tells LaTeX to insert a normal interword space.

The solution to the second problem is to *precede* the period with the command `\@`, which indicates that it is a sentence-ending period.

The corrected examples look as follows:

```
Where is Mr.\ Rogers' neighborhood?
Free software operating systems (e.g.\ GNU/Linux) are great!
I like the poetry of e.\ e.\ cummings.

You know better than I\@.
My middle initial is K\@.
The correct answer to the sample question is C\@.
```

Remember that these rules apply to exclamation points, question marks, and colons as well.

Although we will deal with controlling line breaks in more detail later, it's worth noting now that LaTeX provides a non-breaking space. This produces an interword space and tells LaTeX not to break the line between the two words that the space separates. The symbol for a non-breaking space is '`~`'. That's why it's a special character! It's most often used in situations like the following:

```
Have you read any works by J.~R.~R.~Tolkien?
Chapter~2 is the most exciting part of the book.
```

4.4 Perfect Punctuation

In the previous section we saw how you must occasionally be careful when using periods, exclamation points, question marks, and colons, though otherwise their use is just as you would expect. Commas, semicolons, and apostrophes are typed just as you would expect.

Quotation marks, dashes, and hyphens are different, and they are the subject of this section. These symbols appear differently in printed text than they do on a typewriter or in a text file, so LaTeX handles them differently.

4.4.1 Quotation Marks

Beginning and ending quotation marks look different in printed text, so LaTeX requires that you type them differently. A single beginning quotation mark is made by the character '`` ` ``', which is one of those keys that doesn't have a standard place on the keyboard. A single final quotation mark is made by the character '`'`', which is also used as an apostrophe. For double quotes, just type each quote mark twice. Consider the following examples:

```
The advertisement included the phrase `Satisfaction Guaranteed'.
``What's going on here?''
```

Note that the ugly symmetric double quotation mark key '`"`' is not needed for plain text. If used in normal text, it produces a final double quotation mark. Of course, this symbol may still be needed if your document includes examples of computer code or keyboard input.

Typing the "backwards" quotation marks takes a little getting used to, but tends to become very natural. If you use Emacs to edit your input files, then you probably don't have to worry about it much. One of the features of Emacs' LaTeX mode is that when you type `"`, it guesses (usually correctly) whether this is the beginning or end of a quotation and inserts '` `` `' or '`''`' accordingly. Note that this doesn't work for single quotes, because the end quote '`'`' is also the apostrophe.

4.4.2 Dashes

There are three sizes of dashes in most printed text. In order of increasing size, they are a hyphen, an en-dash, and an em-dash. Hyphens are short dashes that join two words, such as part-time. Hyphens appear in LaTeX input as the character '`-`'. Note that hyphens are also used when a word is split across two lines. That use of the hyphen is covered in the next section.

En-dashes are longer than hyphens and are typically used to indicate a range of numbers, such as 17–23. N-dashes are indicated to LaTeX by two consecutive hyphens: '`--`'.

Em-dashes are longer than en-dashes and are typically used to separate clauses in sentences. They are indicated to LaTeX by three consecutive hyphens: '`---`'.

Consider the following examples:

```
Are you looking for part-time or full-time employment?
The article appears on pages 137--153.
Certain people---you know who I'm talking about---can be annoying.
```

In general, no space occurs before or after any of these dashes.

4.5 Handling Hyphens

(Somebody needs to complete and correct this.)

The explicit use of hyphens in words like part-time is covered in the previous section. In this section we consider the use of hyphens when a word is broken over two lines.

LaTeX has automatic ways of determining where to hyphenate, so most of the time it isn't necessary for you to think about it. However, LaTeX may make mistakes on proper nouns or foreign words. One solution is for you to indicate *soft hyphens* in the input file. Soft hyphens tell LaTeX where a word may be broken, but they do not appear in the output unless the word is actually broken at that point.

Soft hyphens are indicated with the command `\-` in the middle of a word. For example, '`super\-cali\-fraj\-ilistic\-expeala\-docious`'. If any soft hyphens appear in a word, then LaTeX will not break that word at any other points, even if its own rules indicate that it could.

(How do you turn hyphenation off completely?)

4.6 Fonts and Typefaces

There are several aspects of the font that you can control. These include the family (roman, sans serif, typewriter), the shape (upright, italic, slanted, small caps), the series (medium, bold), and the size. You can also switch to an entirely different font, but we won't cover that in this section. (??? add a cross reference)

There are three ways to control the family, shape, and series. One option, especially for small amounts of text, is a command form in which the text is given as the argument for the command. The following list gives those commands.

`\textrm{}`
: Roman family (default).

`\textsf{}`
: Sans serif family.

`\texttt{}`
: Typewriter family.

`\textup{}`
: Upright shape (default).

`\textit{}`
: Italic shape.

`\textsl{}`
: Slanted shape (not the same as italic).

`\textsc{}`
: Small caps shape.

`\textmd{}`
: Medium weight series (default).

`\textbf{}`
: Boldface series.

The two other ways to control these aspects of a font are with declarations (see Section 3.1.4 [Declarations and Scope], page 13) and environments (see Section A.6 [Environments], page 42). In each case the environment name is the same as the declaration but without the '\'. The following list gives the declarations.

`\rmfamily`
: Roman family (default).

`\sffamily`
: Sans serif family.

`\ttfamily`
: Typewriter family.

`\upshape` Upright shape (default).

`\itshape` Italic shape.

`\slshape` Slanted shape (not the same as italic).

`\scshape` Small caps shape.

`\mdseries`
: Medium weight series (default).

`\bfseries`
: Boldface series.

There are also a few additional commands that don't quite fit in the categories above:

`\textnormal{}`
: Roman, upright, medium weight.

`\normalfont`
: Declaration form and environment name equivalent to `\textnormal`.

`\emph{}` Emphasis. Switches between `\textit` and `\textrm`.

`\em` Declaration form and environment name equivalent to `\emph`.

In many cases, the size of the text is handled automatically by LaTeX. For example, section headings are bigger than normal text, so you don't need to worry about it. If you want to change the default text size for the entire document, use an option for the `\documentclass` command instead. (??? add a xref)

There are times when you may want to control the size of the text yourself. The following list shows declarations that allow you to do so. As before, the same names (without the '\') serve as environment names. Carefully note the capitalization of these commands. In some cases all letters are lowercase, in some the first letter is capitalized, and in others all letters are capitalized.

- \tiny
- \scriptsize
- \footnotesize
- \small
- \normalsize
- \large
- \Large
- \LARGE
- \huge
- \Huge

4.7 Foreign Languages

Using LaTeX for languages that use a Latin-based alphabet is relatively straightforward. In most cases, the primary differences are accents and possibly a few additional letters and/or punctuation marks.

These sections cover the basic techniques as well as specifics for French, German, and Spanish.

(??? Someone should add something about capabilities for languages based on other alphabets.)

4.7.1 General Foreign Typesetting

When entirely separate letters or punctuation marks are offered, they are indicated with a LaTeX command. A number of these commands are given below.

`\textexclamdown`	Upside down exclamation point
`\textquestiondown`	Upside down question mark
`\AE \ae`	ae ligature
`\OE \oe`	oe ligature
`\O \o`	'`O`' with a slash through it
`\i`	Dotless '`i`'
`\j`	Dotless '`j`'
`\ss`	sz ligature (German es-tset)

When these commands appear in the middle of the word it is necessary to indicate the end of the command somehow. Although this may be done with a space (which won't appear in the output), it often looks better (in the input) to surround the command with braces. For example: '`verkt\o y`' and '`verkt{\o}y`' should produce the same one-word output.

Many types of accents may also be indicated by using a command. The following commands produce accents on the first letter of their argument. They may also be used without an argument, in which case they operate on the letter immediately following the command. Most of the accents are easy to remember since the accent looks like the character in the command.

`` \` ``	Grave
`\'`	Acute
`\^`	Circumflex
`\~`	Tilde
`\"`	Umlaut (two dots)
`\H`	Two slanted lines (almost an umlaut)
`\r`	Circle
`\v`	Small vee
`\u`	Caron (half circle)
`\=`	Straight line
`\.`	Single dot

In addition, there are three modifications that appear below the letter.

`\b`	Straight line

`\c`	Cedilla
`\d`	Single dot

The commands that are letters need to be followed by space or enclosed in braces as discussed earlier.

The dotless '`i`' and dotless '`j`' are often used when an accent is placed over an '`i`' or '`j`', since otherwise the normal dot will still be present. For example,

```
As{\'\i} es la vida.
```

They may also be used by themselves. For example, the Turkish alphabet includes both dotted and dotless '`i`'.

4.7.2 Typesetting French

In French, the acute `\'`, grave `` \` ``, and circumflex `\^` accents are common over many vowels. The cedilla is often used with a '`c`': `\c{c}`.

For example,

```
Fran\c{c}ois, o\'u \'est l'h\^opital?
```

4.7.3 Typesetting German

German uses the umlaut `\"` over several vowels, and adds the extra letter es-tset `\ss`. For example,

```
Ich bin h\"a{\ss}lich.
```

4.7.4 Typesetting Spanish

Spanish typically only uses an acute accent `\'`. When this occurs over an '`i`', a dotless '`i`' should be used: `\'\i`. In addition a tilde over an '`n`' forms an additional letter: `\~n`. The inverted exclamation point `\textexclamdown` and inverted question mark `\textquestiondown` are also used.

For example,

```
\textquestiondown D\'onde est\'a el ba\~no?
```

There may be shorter commands for the inverted punctuation. Otherwise it's probably worthwhile defining a shorter command yourself (add a reference here).

5 Logical Document Structure

5.1 Classes of Documents

5.2 Chapters and Sections

5.3 Figures

5.4 References

5.5 Citations

5.6 Making a Table of Contents

5.7 Appendicies

5.8 Indicies

5.9 Bibliography

6 Document Layout

6.1 Lists

6.2 Tabular Environment

6.3 Quote and Quotation Environments

6.4 Verbatim Environment

6.5 Verse Environment

6.6 Drawing Images

6.7 Including Images

6.8 Page Layout

7 Mathematics

7.1 Basic Math in LaTeX

7.2 Display Math

7.3 Theorems and Equations

7.4 Elaborate Math Typesetting

8 Computer Programs and Documentation

8.1 Basic Source Code Typesetting

8.2 Defining Modules

8.3 Declaring Functions

8.4 Variables and Constants

8.5 Literate Programming

9 Customizing LaTeX

9.1 Documents

9.2 Fonts

9.3 Custom Lists

9.4 Lengths and Measures

10 Packages, Commands, and Internals

10.1 Loading Packages

10.2 Writing Commands

10.3 Writing Packages

10.4 LaTeX Internals

10.5 LaTeX Programs

Appendix A Command Reference

A LaTeX command begins with the command name, which consists of a `\` followed by either a string of letters or a single non-letter. Arguments contained in square brackets, `[]`, are optional while arguments contained in braces, `{}`, are required.

Note: LaTeX is case sensitive. Enter all commands in lower case unless explicitly directed to do otherwise.

A.1 Counters

Everything LaTeX numbers for you has a counter associated with it. The name of the counter is the same as the name of the environment or command that produces the number, except with no `\`. (enumi - enumiv are used for the nested enumerate environment.) Below is a list of the counters used in LaTeX's standard document classes to control numbering.

`part`	`paragraph`	`figure`	`enumi`
`chapter`	`subparagraph`	`table`	`enumii`
`section`	`page`	`footnote`	`enumiii`
`subsection`	`equation`	`mpfootnote`	`enumiv`
`subsubsection`			

A.1.1 \addtocounter

`\addtocounter{counter}{value}`

The `\addtocounter` command increments the `counter` by the amount specified by the `value` argument. The `value` argument can be negative.

A.1.2 \alph

`\alph{counter}`

This command causes the value of the `counter` to be printed in alphabetic characters. The `\alph` command uses lower case alphabetic alphabetic characters, i.e., `a, b, c...` while the `\Alph` command uses upper case alphabetic characters, i.e., `A, B, C...`.

A.1.3 \arabic

`\arabic{counter}`

The `\arabic` command causes the value of the `counter` to be printed in Arabic numbers, i.e., `3`.

A.1.4 \fnsymbol

`\fnsymbol{counter}`

The `\fnsymbol` command causes the value of the `counter` to be printed in a specific sequence of nine symbols that can be used for numbering footnotes.

eg. From 1-9: $* \dagger \ddagger \S \P \| ** \dagger\dagger \ddagger\ddagger$

NB. `counter` must have a value between 1 and 9 inclusive.

A.1.5 \newcounter

`\newcounter{foo}[counter]`

The `\newcounter` command defines a new counter named `foo`. The counter is initialized to zero.

The optional argument `[counter]` causes the counter `foo` to be reset whenever the counter named in the optional argument is incremented.

A.1.6 \refstepcounter

`\refstepcounter{counter}`

The `\refstepcounter` command works like `\stepcounter`, except it also defines the current `\ref` value to be the result of `\thecounter`.

A.1.7 \roman

`\roman{counter}`

This command causes the value of the `counter` to be printed in Roman numerals. The `\roman` command uses lower case Roman numerals, i.e., `i, ii, iii...`, while the `\Roman` command uses upper case Roman numerals, i.e., `I, II, III...`.

A.1.8 \stepcounter

`\stepcounter{counter}`

The `\stepcounter` command adds one to the `counter` and resets all subsidiary counters.

A.1.9 \setcounter

`\setcounter{counter}{value}`

The `\setcounter` command sets the value of the `counter` to that specified by the `value` argument.

A.1.10 \usecounter

`\usecounter{counter}`

The `\usecounter` command is used in the second argument of the `list` environment to allow the counter specified to be used to number the list items.

A.1.11 \value

`\value{counter}`

The `\value` command produces the value of the `counter` named in the mandatory argument. It can be used where LaTeX expects an integer or number, such as the second argument of a `\setcounter` or `\addtocounter` command, or in:

```
\hspace{\value{foo}\parindent}
```

It is useful for doing arithmetic with counters.

A.2 Cross References

One reason for numbering things like figures and equations is to refer the reader to them, as in "See Figure 3 for more details."

A.2.1 \label

`\label{key}`

A `\label` command appearing in ordinary text assigns to the `key` the number of the current sectional unit; one appearing inside a numbered environment assigns that number to the `key`.

A `key` can consist of any sequence of letters, digits, or punctuation characters. Upper and lowercase letters are different.

To avoid accidentally creating two labels with the same name, it is common to use labels consisting of a prefix and a suffix separated by a colon. The prefixes conventionally used are

- `cha` for chapters
- `sec` for lower-level sectioning commands
- `fig` for figures
- `tab` for tables
- `eq` for equations

Thus, a label for a figure would look like `fig:bandersnatch`.

A.2.2 \pageref

`\pageref{key}`

The `\pageref` command produces the page number of the place in the text where the corresponding `\label` command appears. ie. where `\label{key}` appears.

A.2.3 \ref

`\ref{key}`

The `\ref` command produces the number of the sectional unit, equation number, ... of the corresponding `\label` command.

A.3 Definitions

A.3.1 \newcommand

```
\newcommand{cmd}[args]{definition}
\newcommand{cmd}[args][default]{definition}
\renewcommand{cmd}[args]{definition}
\renewcommand{cmd}[args][default]{definition}
```

These commands define (or redefine) a command.

`cmd`	A command name beginning with a `\`. For `\newcommand` it must not be already defined and must not begin with `\end`; for `\renewcommand` it must already be defined.
`args`	An integer from 1 to 9 denoting the number of arguments of the command being defined. The default is for the command to have no arguments.
`def`	If this optional parameter is present, it means that the command's first argument is optional. The default value of the optional argument is `def`.
`definition`	The text to be substituted for every occurrence of `cmd`; a parameter of the form `#n` in `cmd` is replaced by the text of the nth argument when this substitution takes place.

A.3.2 \newenvironment

```
\newenvironment{nam}[args]{begdef}{enddef}
\newenvironment{nam}[args][default]{begdef}{enddef}
\renewenvironment{nam}[args]{begdef}{enddef}
```

These commands define or redefine an environment.

`nam`	The name of the environment. For `\newenvironment` there must be no currently defined environment by that name, and the command `\nam` must be undefined. For `\renewenvironment` the environment must already be defined.
`args`	An integer from 1 to 9 denoting the number of arguments of the newly-defined environment. The default is no arguments.
`default`	If this is specified, the first argument is optional, and `default` gives the default value for that argument.
`begdef`	The text substituted for every occurrence of `\begin{nam}`; a parameter of the form `#n` in `cmd` is replaced by the text of the nth argument when this substitution takes place.
`enddef`	The text substituted for every occurrence of `\end{nam}`. It may not contain any argument parameters.

A.3.3 \newtheorem

```
\newtheorem{env_name}{caption}[within]
\newtheorem{env_name}[numbered_like]{caption}
```

This command defines a theorem-like environment.

`env_name` The name of the environment to be defined. A string of letters. It must not be the name of an existing environment or counter.

`caption` The text printed at the beginning of the environment, right before the number. This may simply say "Theorem", for example.

`within` The name of an already defined counter, usually of a sectional unit. Provides a means of resetting the new theorem counter **within** the sectional unit.

`numbered_like`
The name of an already defined theorem-like environment.

The `\newtheorem` command may have at most one optional argument.

A.3.4 \newfont

`\newfont{cmd}{font_name}`

Defines the command name `cmd`, which must not be currently defined, to be a declaration that selects the font named `font_name` to be the current font.

A.4 Document Classes

Valid LaTeX document classes include:

- article
- report
- letter
- book
- slides

Other document classes are often available. See Chapter 1 [Overview], page 1, for details. They are selected with the following command:

```
\documentclass [options] {class}
```

All the standard classes (except slides) accept the following options for selecting the typeface size (10 pt is default):

10pt, 11pt, 12pt

All classes accept these options for selecting the paper size (default is letter):

a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper

Miscellaneous options:

- landscape — selects landscape format. Default is portrait.
- titlepage, notitlepage — selects if there should be a separate title page.
- leqno — equation number on left side of equations. Default is right side.
- fleqn — displayed formulas flush left. Default is centred.
- openbib — use “open” bibliography format.
- draft, final — mark/do not mark overfull boxes with a rule. Default is final.

These options are not available with the slides class:

- oneside, twoside — selects one- or twosided layout. Default is oneside, except for the book class.
- openright, openany — determines if a chapter should start on a right-hand page. Default is openright for book.
- onecolumn, twocolumn — one or two columns. Defaults to one column.

The slides class offers the option `clock` for printing the time at the bottom of each note.

If you specify more than one option, they must be separated by a comma.

Additional packages are loaded by a

```
\usepackage[options]{pkg}
```

command. If you specify more than one package, they must be separated by a comma.

Any options given in the `\documentclass` command that are unknown by the selected document class are passed on to the packages loaded with `\usepackage`.

A.5 Layout

Miscellaneous commands for controlling the general layout of the page.

A.5.1 \flushbottom

The `\flushbottom` declaration makes all text pages the same height, adding extra vertical space when necessary to fill out the page.

This is the standard if twocolumn mode is selected.

A.5.2 \onecolumn

The `\onecolumn` declaration starts a new page and produces single-column output.

A.5.3 \raggedbottom

The `\raggedbottom` declaration makes all pages the height of the text on that page. No extra vertical space is added.

A.5.4 \twocolumn

```
\twocolumn[text]
```

The `\twocolumn` declaration starts a new page and produces two-column output. If the optional `text` argument is present, it is typeset in one-column mode.

A.6 Environments

LaTeX provides a number of different paragraph-making environments. Each environment begins and ends in the same manner.

```
\begin{environment-name}
.
.
.
\end{environment-name}
```

A.6.1 array

```
\begin{array}{col1col2...coln}
column 1 entry & column 2 entry ... & column n entry \\
 .
 .
 .
\end{array}
```

Math arrays are produced with the array environment. It has a single mandatory argument describing the number of columns and the alignment within them. Each column, `coln`, is specified by a single letter that tells how items in that row should be formatted.

- `c` — for centred
- `l` — for flush left
- `r` — for flush right

Column entries must be separated by an `&`. Column entries may include other LaTeX commands. Each row of the array must be terminated with the string `\\`.

Note that the `array` environment can only be used in math mode, so normally it is used inside an `equation` environment.

A.6.2 center

```
\begin{center}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{center}
```

The `center` environment allows you to create a paragraph consisting of lines that are centred within the left and right margins on the current page. Each line must be terminated with the string `\\`.

A.6.2.1 \centering

This declaration corresponds to the `center` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`. The text of a figure or table can be centred on the page by putting a `\centering` command at the beginning of the figure or table environment.

Unlike the `center` environment, the `\centering` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

A.6.3 description

```
\begin{description}
\item [label] First item
\item [label] Second item
.
.
.
\end{description}
```

The `description` environment is used to make labelled lists. The `label` is bold face and flushed right.

A.6.4 enumerate

```
\begin{enumerate}
\item First item
\item Second item
.
.
.
\end{enumerate}
```

The `enumerate` environment produces a numbered list. Enumerations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an enumerated list begins with an `\item` command. There must be at least one `\item` command within the environment.

The `enumerate` environment uses the `enumi` through `enumiv` counters (see Section A.1 [Counters], page 35). The type of numbering can be changed by redefining `\theenumi` etc.

A.6.5 eqnarray

```
\begin{eqnarray}
math formula 1 \\
math formula 2 \\
.
.
.
\end{eqnarray}
```

The `eqnarray` environment is used to display a sequence of equations or inequalities. It is very much like a three-column `array` environment, with consecutive rows separated by `\\` and consecutive items within a row separated by an `&`.

An equation number is placed on every line unless that line has a `\nonumber` command.

The command `\lefteqn` is used for splitting long formulas across lines. It typesets its argument in display style flush left in a box of zero width.

A.6.6 equation

```
\begin{equation}
 math formula
\end{equation}
```

The `equation` environment centres your equation on the page and places the equation number in the right margin.

A.6.7 figure

```
\begin{figure}[placement]

 body of the figure

\caption{figure title}
\end{figure}
```

Figures are objects that are not part of the normal text, and are usually “floated” to a convenient place, like the top of a page. Figures will not be split between two pages.

The optional argument `[placement]` determines where LaTeX will try to place your figure. There are four places where LaTeX can possibly put a float:

1. `h` (Here) - at the position in the text where the figure environment appears.
2. `t` (Top) - at the top of a text page.
3. `b` (Bottom) - at the bottom of a text page.
4. `p` (Page of floats) - on a separate float page, which is a page containing no text, only floats.

The standard report and article classes use the default placement `tbp`.

The body of the figure is made up of whatever text, LaTeX commands, etc. you wish. The `\caption` command allows you to title your figure.

A.6.8 flushleft

```
\begin{flushleft}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushleft}
```

The `flushleft` environment allows you to create a paragraph consisting of lines that are flushed left, to the left-hand margin. Each line must be terminated with the string `\\`.

A.6.8.1 \raggedright

This declaration corresponds to the `flushleft` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushleft` environment, the `\raggedright` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

A.6.9 flushright

```
\begin{flushright}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushright}
```

The `flushright` environment allows you to create a paragraph consisting of lines that are flushed right, to the right-hand margin. Each line must be terminated with the string `\\`.

A.6.9.1 \raggedleft

This declaration corresponds to the `flushright` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushright` environment, the `\raggedleft` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

A.6.10 itemize

```
\begin{itemize}
\item First item
\item Second item
.
.
.
\end{itemize}
```

The `itemize` environment produces a "bulleted" list. Itemizations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an `itemized` list begins with an `\item` command. There must be at least one `\item` command within the environment.

The `itemize` environment uses the `itemi` through `itemiv` counters (see Section A.1 [Counters], page 35). The type of numbering can be changed by redefining `\theitemi` etc.

A.6.11 letter

This environment is used for creating letters. See Section A.9 [Letters], page 58.

A.6.12 list

The `list` environment is a generic environment which is used for defining many of the more specific environments. It is seldom used in documents, but often in macros.

```
\begin{list}{label}{spacing}
\item First item
\item Second item
.
.
.
\end{list}
```

The `{label}` argument specifies how items should be labelled. This argument is a piece of text that is inserted in a box to form the label. This argument can and usually does contain other LaTeX commands.

The `{spacing}` argument contains commands to change the spacing parameters for the list. This argument will most often be null, i.e., `{}`. This will select all default spacing which should suffice for most cases.

A.6.13 minipage

```
\begin{minipage}[position]{width}
 text
\end{minipage}
```

The `minipage` environment is similar to a `\parbox` command. It takes the same optional `position` argument and mandatory `width` argument. You may use other paragraph-making environments inside a minipage.

Footnotes in a `minipage` environment are handled in a way that is particularly useful for putting footnotes in figures or tables. A `\footnote` or `\footnotetext` command puts the footnote at the bottom of the minipage instead of at the bottom of the page, and it uses the `mpfootnote` counter instead of the ordinary `footnote` counter See Section A.1 [Counters], page 35.

NOTE: Don't put one minipage inside another if you are using footnotes; they may wind up at the bottom of the wrong minipage.

A.6.14 picture

```
\begin{picture}(width,height)(x offset,y offset)
  .
  .
 picture commands
  .
  .
\end{picture}
```

The `picture` environment allows you to create just about any kind of picture you want containing text, lines, arrows and circles. You tell LaTeX where to put things in the picture by specifying their coordinates. A coordinate is a number that may have a decimal point and a minus sign — a number like `5`, `2.3` or `-3.1416`. A coordinate specifies a length in multiples of the unit length `\unitlength`, so if `\unitlength` has been set to `1cm`, then the coordinate 2.54 specifies a length of 2.54 centimetres. You can change the value of `\unitlength` anywhere you want, using the `\setlength` command, but strange things will happen if you try changing it inside the picture environment.

A position is a pair of coordinates, such as `(2.4,-5)`, specifying the point with x-coordinate `2.4` and y-coordinate `-5`. Coordinates are specified in the usual way with respect to an origin, which is normally at the lower-left corner of the picture. Note that when a position appears as an argument, it is not enclosed in braces; the parentheses serve to delimit the argument.

The `picture` environment has one mandatory argument, which is a `position`. It specifies the size of the picture. The environment produces a rectangular box with width and height determined by this argument's x- and y-coordinates.

The `picture` environment also has an optional `position` argument, following the `size` argument, that can change the origin. (Unlike ordinary optional arguments, this argument is not contained in square brackets.) The optional argument gives the coordinates of the point at the lower-left corner of the picture (thereby determining the origin). For example, if `\unitlength` has been set to `1mm`, the command

```
\begin{picture}(100,200)(10,20)
```

produces a picture of width 100 millimetres and height 200 millimetres, whose lower-left corner is the point (10,20) and whose upper-right corner is therefore the point (110,220). When you first draw a picture, you will omit the optional argument, leaving the origin at the lower-left corner. If you then want to modify your picture by shifting everything, you just add the appropriate optional argument.

The environment's mandatory argument determines the nominal size of the picture. This need bear no relation to how large the picture really is; LaTeX will happily allow you to put things outside the picture, or even off the page. The picture's nominal size is used by LaTeX in determining how much room to leave for it.

Everything that appears in a picture is drawn by the `\put` command. The command

```
\put (11.3,-.3){...}
```

puts the object specified by `...` in the picture, with its reference point at coordinates (11.3,-.3). The reference points for various objects will be described below.

The `\put` command creates an *LR box*. You can put anything in the text argument of the `\put` command that you'd put into the argument of an `\mbox` and related commands. When you do this, the reference point will be the lower left corner of the box.

Picture commands:

A.6.14.1 \circle

`\circle[*]{diameter}`

The `\circle` command produces a circle with a diameter as close to the specified one as possible. If the `*`-form of the command is used, LaTeX draws a solid circle.

Note that only circles up to 40 pt can be drawn.

A.6.14.2 \dashbox

Draws a box with a dashed line.

`\dashbox{dash_length}(width,height){...}`

The `\dashbox` has an extra argument which specifies the width of each dash. A dashed box looks best when the `width` and `height` are multiples of the `dash_length`.

A.6.14.3 \frame

`\frame{...}`

The `\frame` command puts a rectangular frame around the object specified in the argument. The reference point is the bottom left corner of the frame. No extra space is put between the frame and the object.

A.6.14.4 \framebox

`\framebox(width,height)[position]{...}`

The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The `framebox` command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

A.6.14.5 \line

`\line(x slope,y slope){length}`

The `\line` command draws a line of the specified `length` and `slope`.

Note that LaTeX can only draw lines with slope = x/y, where x and y have integer values from -6 through 6.

A.6.14.6 \linethickness

`\linethickness{dimension}`

Declares the thickness of horizontal and vertical lines in a picture environment to be `dimension`, which must be a positive length. It does not affect the thickness of slanted lines and circles, or the quarter circles drawn by `\oval` to form the corners of an oval.

A.6.14.7 \makebox

`\makebox(width,height)[position]{...}`

The `\makebox` command for the picture environment is similar to the normal `\makebox` command except that you must specify a `width` and `height` in multiples of `\unitlength`.

The optional argument, `[position]`, specifies the quadrant that your text appears in. You may select up to two of the following:

- `t` - Moves the item to the top of the rectangle
- `b` - Moves the item to the bottom
- `l` - Moves the item to the left
- `r` - Moves the item to the right

A.6.14.8 \multiput

`\multiput(x coord,y coord)(delta x,delta y){number of copies}{object}`

The `\multiput` command can be used when you are putting the same object in a regular pattern across a picture.

A.6.14.9 \oval

`\oval(width,height)[portion]`

The `\oval` command produces a rectangle with rounded corners. The optional argument, `[portion]`, allows you to select part of the oval.

- `t` - Selects the top portion
- `b` - Selects the bottom portion
- `r` - Selects the right portion
- `l` - Selects the left portion

Note that the “corners” of the oval are made with quarter circles with a maximum radius of 20 pt, so large “ovals” will look more like boxes with rounded corners.

A.6.14.10 \put

`\put(x coord,y coord){ ... }`

The `\put` command places the item specified by the mandatory argument at the given coordinates.

A.6.14.11 \shortstack

`\shortstack[position]{... \\ ... \\ ...}`

The `\shortstack` command produces a stack of objects. The valid positions are:

- `r` - Moves the objects to the right of the stack
- `l` - Moves the objects to the left of the stack
- `c` - Moves the objects to the centre of the stack (default)

A.6.14.12 \vector

`\vector(x slope,y slope){length}`

The `\vector` command draws a line with an arrow of the specified length and slope. The `x` and `y` values must lie between -4 and +4, inclusive.

A.6.15 quotation

```
\begin{quotation}
 text
\end{quotation}
```

The margins of the `quotation` environment are indented on the left and the right. The text is justified at both margins and there is paragraph indentation. Leaving a blank line between text produces a new paragraph.

A.6.16 quote

```
\begin{quote}
 text
\end{quote}
```

The margins of the `quote` environment are indented on the left and the right. The text is justified at both margins. Leaving a blank line between text produces a new paragraph.

A.6.17 tabbing

```
\begin{tabbing}
text \= more text \= still more text \= last text \\
second row \>  \> more \\
.
.
.
\end{tabbing}
```

The `tabbing` environment provides a way to align text in columns. It works by setting tab stops and tabbing to them much the way you do with an ordinary typewriter.

It is best suited for cases where the width of each column is constant and known in advance.

This environment can be broken across pages, unlike the `tabular` environment.

The following commands can be used inside a `tabbing` enviroment:

`\=`

Sets a tab stop at the current position.

`\>`

Advances to the next tab stop.

`\<`

This command allows you to put something to the left of the local margin without changing the margin. Can only be used at the start of the line.

`\+`

Moves the left margin of the next and all the following commands one tab stop to the right.

`\-`

Moves the left margin of the next and all the following commands one tab stop to the left.

`\'`

Moves everything that you have typed so far in the current column, i.e. everything from the most recent `\>`, `\<`, `\'`, `\\`, or `\kill` command, to the right of the previous column, flush against the current column's tab stop.

``\` ``

Allows you to put text flush right against any tab stop, including tab stop 0. However, it can't move text to the right of the last column because there's no tab stop there. The ``\` `` command moves all the text that follows it, up to the `\\` or `\end{tabbing}` command that ends the line, to the right margin of the tabbing environment. There must be no `\>` or `\'` command between the ``\` `` and the command that ends the line.

`\kill`

Sets tab stops without producing text. Works just like `\\` except that it throws away the current line instead of producing output for it. The effect of any `\=`, `\+` or `\-` commands in that line remain in effect.

`\pushtabs`
: Saves all current tab stop positions. Useful for temporarily changing tab stop positions in the middle of a `tabbing` environment.

`\pushtabs`
: Restores the tab stop positions saved by the last `\pushtabs`.

`\a`
: In a `tabbing` environment, the commands `\=`, `\'` and `\`` do not produce accents as normal. Instead, the commands `\a=`, `\a'` and `\a`` are used.

This example typesets a Pascal function in a traditional format:

```
\begin{tabbing}
function \= fact(n : integer) : integer;\\
         \> begin \= \+ \\
               \> if \= n $>$ 1 then \+ \\
                        fact := n * fact(n-1) \- \\
                  else \+ \\
                        fact := 1; \-\- \\
            end;\\
\end{tabbing}
```

A.6.18 table

```
\begin{table}[placement]

 body of the table

\caption{table title}
\end{table}
```

Tables are objects that are not part of the normal text, and are usually “floated” to a convenient place, like the top of a page. Tables will not be split between two pages.

The optional argument `[placement]` determines where LaTeX will try to place your table. There are four places where LaTeX can possibly put a float:

- `h` : Here - at the position in the text where the table environment appears.
- `t` : Top - at the top of a text page.
- `b` : Bottom - at the bottom of a text page.
- `p` : Page of floats - on a separate float page, which is a page containing no text, only floats.

The standard `report` and `article` classes use the default placement `[tbp]`.

The body of the table is made up of whatever text, LaTeX commands, etc., you wish. The `\caption` command allows you to title your table.

A.6.19 tabular

```
\begin{tabular}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \\
.
```

```
.
.
\end{tabular}
```

or

```
\begin{tabular*}{width}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \\
.
.
.
\end{tabular*}
```

These environments produce a box consisting of a sequence of rows of items, aligned vertically in columns. The mandatory and optional arguments consist of:

`width` Specifies the width of the `tabular*` environment. There must be rubber space between columns that can stretch to fill out the specified width.

`pos` Specifies the vertical position; default is alignment on the centre of the environment.

- `t` - align on top row
- `b` - align on bottom row

`cols` Specifies the column formatting. It consists of a sequence of the following specifiers, corresponding to the sequence of columns and intercolumn material.

- `l` - A column of left-aligned items.
- `r` - A column of right-aligned items.
- `c` - A column of centred items.
- `|` - A vertical line the full height and depth of the environment.
- `@{text}` - This inserts `text` in every row. An `@`-expression suppresses the intercolumn space normally inserted between columns; any desired space between the inserted text and the adjacent items must be included in text. An `\extracolsep{wd}` command in an `@`-expression causes an extra space of width `wd` to appear to the left of all subsequent columns, until countermanded by another `\extracolsep` command. Unlike ordinary intercolumn space, this extra space is not suppressed by an `@`-expression. An `\extracolsep` command can be used only in an `@`-expression in the `cols` argument.
- `p{wd}` - Produces a column with each item typeset in a parbox of width `wd`, as if it were the argument of a `\parbox[t]{wd}` command. However, a `\\` may not appear in the item, except in the following situations:
  1. inside an environment like `minipage`, `array`, or `tabular`.
  2. inside an explicit `\parbox`.
  3. in the scope of a `\centering`, `\raggedright`, or `\raggedleft` declaration. The latter declarations must appear inside braces or an environment when used in a `p`-column element.

- `*{num}{cols}` - Equivalent to `num` copies of `cols`, where `num` is any positive integer and `cols` is any list of column-specifiers, which may contain another `*-expression`.

These commands can be used inside a `tabular` environment:

A.6.19.1 \cline

`\cline{i-j}`

The `\cline` command draws horizontal lines across the columns specified, beginning in column `i` and ending in column `j`, which are identified in the mandatory argument.

A.6.19.2 \hline

The `\hline` command will draw a horizontal line the width of the table. It's most commonly used to draw a line at the top, bottom, and between the rows of the table.

A.6.19.3 \multicolumn

`\multicolumn{cols}{pos}{text}`

The `\multicolumn` is used to make an entry that spans several columns. The first mandatory argument, `cols`, specifies the number of columns to span. The second mandatory argument, `pos`, specifies the formatting of the entry; `c` for centred, `l` for flushleft, `r` for flushright. The third mandatory argument, `text`, specifies what text is to make up the entry.

A.6.19.4 \vline

The `\vline` command will draw a vertical line extending the full height and depth of its row. An `\hfill` command can be used to move the line to the edge of the column. It can also be used in an `@`-expression.

A.6.20 thebibliography

```
\begin{thebibliography}{widest-label}
\bibitem[label]{cite_key}
.
.
.
\end{thebibliography}
```

The `thebibliography` environment produces a bibliography or reference list. In the `article` class, this reference list is labelled "References"; in the `report` class, it is labelled "Bibliography".

- `widest-label`: Text that, when printed, is approximately as wide as the widest item label produces by the `\bibitem` commands.

A.6.20.1 \bibitem

`\bibitem[label]{cite_key}`

The `\bibitem` command generates an entry labelled by `label`. If the `label` argument is missing, a number is generated as the `label`, using the `enumi` counter. The `cite_key` is any sequence of letters, numbers, and punctuation symbols not containing a comma. This command writes an entry on the '`.aux`' file containing `cite_key` and the item's `label`. When this '`.aux`' file is read by the `\begin{document}` command, the item's `label` is associated with `cite_key`, causing the reference to `cite_key` by a `\cite` command to produce the associated `label`.

A.6.20.2 \cite

`\cite[text]{key_list}`

The `key_list` argument is a list of citation keys. This command generates an in-text citation to the references associated with the keys in `key_list` by entries on the '`.aux`' file read by the `\begin{document}` command.

The optional `text` argument will appear after the citation, i.e. `\cite[p. 2]{knuth}` might produce '[Knuth, p. 2]'.

A.6.20.3 \nocite

`\nocite{key_list}`

The `\nocite` command produces no text, but writes `key_list`, which is a list of one or more citation keys, on the '`.aux`' file.

A.6.20.4 Using BibTeX

If you use the BibTeX program by Oren Patashnik (highly recommended if you need a bibliography of more than a couple of titles) to maintain your bibliography, you don't use the `thebibliography` environment. Instead, you include the lines

```
\bibliographystyle{style}
\bibliography{bibfile}
```

where `style` refers to a file `style.bst`, which defines how your citations will look. The standard styles distributed with BibTeX are:

`alpha`	Sorted alphabetically. Labels are formed from name of author and year of publication.
`plain`	Sorted alphabetically. Labels are numeric.
`unsrt`	Like `plain`, but entries are in order of citation.
`abbrv`	Like `plain`, but more compact labels.

In addition, numerous other BibTeX style files exist tailored to the demands of various publications.

The argument to `\bibliography` refers to the file `bibfile.bib`, which should contain your database in BibTeX format. Only the entries referred to via `\cite` and `\nocite` will be listed in the bibliography.

A.6.21 theorem

```
\begin{theorem}
 theorem text
\end{theorem}
```

The `theorem` environment produces "Theorem x" in boldface followed by your theorem text.

A.6.22 titlepage

```
\begin{titlepage}
 text
\end{titlepage}
```

The `titlepage` environment creates a title page, i.e. a page with no printed page number or heading. It also causes the following page to be numbered page one. Formatting the title page is left to you. The `\today` command comes in handy for title pages.

Note that you can use the `\maketitle` command to produce a standard title page.

A.6.23 verbatim

```
\begin{verbatim}
 text
\end{verbatim}
```

The `verbatim` environment is a paragraph-making environment that gets LaTeX to print exactly what you type in. It turns LaTeX into a typewriter with carriage returns and blanks having the same effect that they would on a typewriter.

A.6.23.1 \verb

`\verb char literal_text char`

`\verb*char literal_text char`

Typesets `literal_text` exactly as typed, including special characters and spaces, using a typewriter (`\tt`) type style. There may be no space between `\verb` or `\verb*` and `char` (space is shown here only for clarity). The `*-form` differs only in that spaces are printed as '\verb*| |'.

A.6.24 verse

```
\begin{verse}
 text
\end{verse}
```

The `verse` environment is designed for poetry, though you may find other uses for it.

The margins are indented on the left and the right. Separate the lines of each stanza with \\, and use one or more blank lines to separate the stanzas.

A.7 Footnotes

Footnotes can be produced in one of two ways. They can be produced with one command, the `\footnote` command. They can also be produced with two commands, the `\footnotemark` and the `\footnotetext` commands. See the specific command for information on why you would use one over the other.

A.7.1 \footnote

```
\footnote[number]{text}
```

The `\footnote` command places the numbered footnote `text` at the bottom of the current page. The optional argument, `number`, is used to change the default footnote number. This command can only be used in outer paragraph mode; i.e., you cannot use it in sectioning commands like `\chapter`, in figures, tables or in a `tabular` environment.

A.7.2 \footnotemark

The `\footnotemark` command puts the footnote `number` in the text. This command can be used in inner paragraph mode. The text of the footnote is supplied by the `\footnotetext` command.

This command can be used to produce several consecutive footnote markers referring to the same footnote by using

```
\footnotemark[\value{footnote}]
```

after the first `\footnote` command.

A.7.3 \footnotetext

```
\footnotetext[number]{text}
```

The `\footnotetext` command produces the `text` to be placed at the bottom of the page. This command can come anywhere after the `\footnotemark` command. The `\footnotetext` command must appear in outer paragraph mode.

The optional argument, `number`, is used to change the default footnote number.

A.8 Lengths

A `length` is a measure of distance. Many LaTeX commands take a length as an argument.

A.8.1 \newlength

`\newlength{\gnat}`

The `\newlength` command defines the mandatory argument, `\gnat`, as a `length` command with a value of `0in`. An error occurs if a `\gnat` command already exists.

A.8.2 \setlength

`\setlength{\gnat}{length}`

The `\setlength` command is used to set the value of a `length` command. The `length` argument can be expressed in any terms of length LaTeX understands, i.e., inches (`in`), millimetres (`mm`), points (`pt`), etc.

A.8.3 \addtolength

`\addtolength{\gnat}{length}`

The `\addtolength` command increments a "length command" by the amount specified in the `length` argument. It can be a negative amount.

A.8.4 \settodepth

`\settodepth{\gnat}{text}`

The `\settodepth` command sets the value of a `length` command equal to the depth of the `text` argument.

A.8.5 \settoheight

`\settoheight{\gnat}{text}`

The `\settoheight` command sets the value of a `length` command equal to the height of the `text` argument.

A.8.6 \settowidth

`\settowidth{\gnat}{text}`

The `\settowidth` command sets the value of a `length` command equal to the width of the `text` argument.

A.8.7 Predefined lengths

`\width`

`\height`

`\depth`

`\totalheight`

These length parameters can be used in the arguments of the box-making commands See Section A.17 [Spaces & Boxes], page 72. They specify the natural width etc. of the text in the box. `\totalheight` equals `\height + \depth`. To make a box with the text stretched to double the natural size, e.g., say

```
\makebox[2\width]{Get a stretcher}
```

A.9 Letters

You can use LaTeX to typeset letters, both personal and business. The `letter` document class is designed to make a number of letters at once, although you can make just one if you so desire.

Your '`.tex`' source file has the same minimum commands as the other document classes, i.e., you must have the following commands as a minimum:

```
\documentclass{letter}
\begin{document}
 ... letters ...
\end{document}
```

Each letter is a `letter` environment, whose argument is the name and address of the recipient. For example, you might have:

```
\begin{letter}{Mr. Joe Smith\\ 2345 Princess St.
     \\ Edinburgh, EH1 1AA}
  ...
\end{letter}
```

The letter itself begins with the `\opening` command. The text of the letter follows. It is typed as ordinary LaTeX input. Commands that make no sense in a letter, like `\chapter`, do not work. The letter closes with a `\closing` command.

After the `closing`, you can have additional material. The `\cc` command produces the usual "cc: ...". There's also a similar `\encl` command for a list of enclosures. With both these commands, use `\\` to separate the items.

These commands are used with the `letter` class:

A.9.1 \address

`\address{Return address}`

The return address, as it should appear on the letter and the envelope. Separate lines of the address should be separated by `\\` commands. If you do not make an `\address` declaration, then the letter will be formatted for copying onto your organisation's standard letterhead. (See Chapter 1 [Overview], page 1, for details on your local implementation). If you give an `\address` declaration, then the letter will be formatted as a personal letter.

A.9.2 \cc

`\cc{Kate Schechter\\Rob McKenna}`

Generate a list of other persons the letter was sent to. Each name is printed on a separate line.

A.9.3 \closing

`\closing{text}`

The letter closes with a `\closing` command, i.e.,

```
\closing{Best Regards,}
```

A.9.4 \encl

```
\encl{CV\\Certificates}
```

Generate a list of enclosed material.

A.9.5 \location

```
\location{address}
```

This modifies your organisation's standard address. This only appears if the `firstpage` pagestyle is selected.

A.9.6 \makelabels

```
\makelabels{number}
```

If you issue this command in the preamble, LaTeX will create a sheet of address labels. This sheet will be output before the letters.

A.9.7 \name

```
\name{June Davenport}
```

Your name, used for printing on the envelope together with the return address.

A.9.8 \opening

```
\opening{text}
```

The letter begins with the `\opening` command. The mandatory argument, `text`, is whatever text you wish to start your letter, i.e.,

```
    \opening{Dear Joe,}
```

A.9.9 \ps

```
\ps
```

Use this command before a postscript.

A.9.10 \signature

```
\signature{Harvey Swick}
```

Your name, as it should appear at the end of the letter underneath the space for your signature. Items that should go on separate lines should be separated by `\\` commands.

A.9.11 \startbreaks

```
\startbreaks
```

Used after a `\stopbreaks` command to allow page breaks again.

A.9.12 \stopbreaks

```
\stopbreaks
```

Inhibit page breaks until a `\startbreaks` command occurs.

A.9.13 \telephone

`\telephone{number}`

This is your telephone number. This only appears if the `firstpage` pagestyle is selected.

A.10 Line & Page Breaking

The first thing LaTeX does when processing ordinary text is to translate your input file into a string of glyphs and spaces. To produce a printed document, this string must be broken into lines, and these lines must be broken into pages. In some environments, you do the line breaking yourself with the `\\` command, but LaTeX usually does it for you.

A.10.1 \\

```
\\[*][extra-space]
```

The `\\` command tells LaTeX to start a new line. It has an optional argument, `extra-space`, that specifies how much extra vertical space is to be inserted before the next line. This can be a negative amount.

The `\\*` command is the same as the ordinary `\\` command except that it tells LaTeX not to start a new page after the line.

A.10.2 \-

The `\-` command tells LaTeX that it may hyphenate the word at that point. LaTeX is very good at hyphenating, and it will usually find all correct hyphenation points. The `\-` command is used for the exceptional cases.

Note that when you insert `\-` commands in a word, the word will only be hyphenated at those points and not at any of the hyphenation points that LaTeX might otherwise have chosen.

A.10.3 \cleardoublepage

The `\cleardoublepage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed. In a two-sided printing style, it also makes the next page a right-hand (odd-numbered) page, producing a blank page if necessary.

A.10.4 \clearpage

The `\clearpage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed.

A.10.5 \enlargethispage

```
\enlargethispage{size}
    \enlargethispage*{size}
```

Enlarge the `\textheight` for the current page by the specified amount; e.g. `\enlargethispage{\baselineskip}` will allow one additional line.

The starred form tries to squeeze the material together on the page as much as possible. This is normally used together with an explicit `\pagebreak`.

A.10.6 \fussy

```
\fussy
```

This declaration (which is the default) makes TeX more fussy about line breaking. This can avoids too much space between words, but may produce overfull boxes.

This command cancels the effect of a previous `\sloppy` command.

A.10.7 \hyphenation

`\hyphenation{words}`

The `\hyphenation` command declares allowed hyphenation points, where `words` is a list of words, separated by spaces, in which each hyphenation point is indicated by a `-` character.

A.10.8 \linebreak

`\linebreak[number]`

The `\linebreak` command tells LaTeX to break the current line at the point of the command. With the optional argument, `number`, you can convert the `\linebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

The `\linebreak` command causes LaTeX to stretch the line so it extends to the right margin.

A.10.9 \newline

The `\newline` command breaks the line right where it is. It can only be used in paragraph mode.

A.10.10 \newpage

The `\newpage` command ends the current page.

A.10.11 \nolinebreak

`\nolinebreak[number]`

The `\nolinebreak` command prevents LaTeX from breaking the current line at the point of the command. With the optional argument, `number`, you can convert the `\nolinebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

A.10.12 \nopagebreak

`\nopagebreak[number]`

The `\nopagebreak` command prevents LaTeX from breaking the current page at the point of the command. With the optional argument, `number`, you can convert the `\nopagebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

A.10.13 \pagebreak

`\pagebreak[number]`

The `\pagebreak` command tells LaTeX to break the current page at the point of the command. With the optional argument, `number`, you can convert the `\pagebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

A.10.14 \sloppy

```
\sloppy
```

This declaration makes TeX less fussy about line breaking. This can prevent overfull boxes, but may leave too much space between words.

Lasts until a `\fussy` command is issued.

A.11 Making Paragraphs

A paragraph is ended by one or more completely blank lines — lines not containing even a `%`. A blank line should not appear where a new paragraph cannot be started, such as in math mode or in the argument of a sectioning command.

A.11.1 \indent

`\indent`

This produces a horizontal space whose width equals the width of the paragraph indentation. It is used to add paragraph indentation where it would otherwise be suppressed.

A.11.2 \noindent

`\noindent`

When used at the beginning of the paragraph, it suppresses the paragraph indentation. It has no effect when used in the middle of a paragraph.

A.11.3 \par

Equivalent to a blank line; often used to make command or environment definitions easier to read.

A.12 Margin Notes

The command `\marginpar[left]{right}` creates a note in the margin. The first line will be at the same height as the line in the text where the `\marginpar` occurs.

When you only specify the mandatory argument `right`, the text will be placed

- in the right margin for one-sided layout
- in the outside margin for two-sided layout
- in the nearest margin for two-column layout.

By issuing the command `\reversemarginpar`, you can force the marginal notes to go into the opposite (inside) margin.

When you specify both arguments, `left` is used for the left margin, and `right` is used for the right margin.

The first word will normally not be hyphenated; you can enable hyphenation by prefixing the first word with a `\hspace{0pt}` command.

A.13 Math Formulae

There are three environments that put LaTeX in math mode:

`math`	For Formulae that appear right in the text.
`displaymath`	For Formulae that appear on their own line.
`equation`	The same as the displaymath environment except that it adds an equation number in the right margin.

The `math` environment can be used in both paragraph and LR mode, but the `displaymath` and `equation` environments can be used only in paragraph mode. The `math` and `displaymath` environments are used so often that they have the following short forms:

```
\(...\)     instead of     \begin{math}...\end{math}

\[...\]     instead of     \begin{displaymath}...\end{displaymath}
```

In fact, the `math` environment is so common that it has an even shorter form:

```
$ ... $     instead of     \(...\)
```

A.13.1 Subscripts & Superscripts

To get an expression *exp* to appear as a subscript, you just type `_{`*exp*`}`. To get *exp* to appear as a superscript, you type `^{`*exp*`}`. LaTeX handles superscripted superscripts and all of that stuff in the natural way. It even does the right thing when something has both a subscript and a superscript.

A.13.2 Math Symbols

LaTeX provides almost any mathematical symbol you're likely to need. The commands for generating them can be used only in math mode. For example, if you include `$\pi$` in your source, you will get the symbol π in your output.

A.13.3 Spacing in Math Mode

In a `math` environment, LaTeX ignores the spaces you type and puts in the spacing that it thinks is best. LaTeX formats mathematics the way it's done in mathematics texts. If you want different spacing, LaTeX provides the following four commands for use in math mode:

1. `\;` - a thick space
2. `\:` - a medium space
3. `\,` - a thin space
4. `\!` - a negative thin space

A.13.4 Math Miscellany

`\cdots` Produces a horizontal ellipsis where the dots are raised to the centre of the line.

eg. $\cdots$

`\ddots` Produces a diagonal ellipsis.

eg. $\ddots$

`\frac{num}{den}`

Produces the fraction `num` divided by `den`.

eg. $\frac{1}{4}$

`\ldots` Produces an ellipsis. This command works in any mode, not just math mode.

eg. $\ldots$

`\overbrace{text}`

Generates a brace over text.

eg. $\overbrace{x+\cdots+x}^{k \text{ times}}$

`\overline{text}`

Causes the argument text to be overlined.

eg. $\overline{x}$

`\sqrt[root]{arg}`

Produces the square root of its argument. The optional argument, `root`, determines what root to produce, i.e., the cube root of `x+y` would be typed as `$\sqrt[3]{x+y}$`.

eg. $\sqrt{x-1}$

`\underbrace{text}`

Generates text with a brace underneath.

eg. $\underbrace{x+y+z}_{>0}$

`\underline{text}`

Causes the argument text to be underlined. This command can also be used in paragraph and LR modes.

eg. $\underline{z}$

`\vdots` Produces a vertical ellipsis.

eg. $\vdots$

A.14 Modes

When LaTeX is processing your input text, it is always in one of three modes:

- Paragraph mode
- Math mode
- Left-to-right mode, called LR mode for short

LaTeX changes mode only when it goes up or down a staircase to a different level, though not all level changes produce mode changes. Mode changes occur only when entering or leaving an environment, or when LaTeX is processing the argument of certain text-producing commands.

"Paragraph mode" is the most common; it's the one LaTeX is in when processing ordinary text. In that mode, LaTeX breaks your text into lines and breaks the lines into pages. LaTeX is in "math mode" when it's generating a mathematical formula. In "LR mode", as in paragraph mode, LaTeX considers the output that it produces to be a string of words with spaces between them. However, unlike paragraph mode, LaTeX keeps going from left to right; it never starts a new line in LR mode. Even if you put a hundred words into an `\mbox`, LaTeX would keep typesetting them from left to right inside a single box, and then complain because the resulting box was too wide to fit on the line.

LaTeX is in LR mode when it starts making a box with an `\mbox` command. You can get it to enter a different mode inside the box - for example, you can make it enter math mode to put a formula in the box. There are also several text-producing commands and environments for making a box that put LaTeX in paragraph mode. The box make by one of these commands or environments will be called a `parbox`. When LaTeX is in paragraph mode while making a box, it is said to be in "inner paragraph mode". Its normal paragraph mode, which it starts out in, is called "outer paragraph mode".

A.15 Page Styles

The `\documentclass` command determines the size and position of the page's head and foot. The page style determines what goes in them.

A.15.1 \maketitle

`\maketitle`

The `\maketitle` command generates a title on a separate title page - except in the `article` class, where the title normally goes at the top of the first page. Information used to produce the title is obtained from the following declarations:

See Section A.15 [Page Styles], page 69, for the commands to give the information.

A.15.2 \author

`\author{names}`

The `\author` command declares the author(s), where `names` is a list of authors separated by `\and` commands. Use `\\` to separate lines within a single author's entry – for example, to give the author's institution or address.

A.15.3 \date

`\date{text}`

The `\date` command declares *text* to be the document's date. With no `\date` command, the current date is used.

A.15.4 \thanks

`\thanks{text}`

The `\thanks` command produces a `\footnote` to the title.

A.15.5 \title

`\title{text}`

The `\title` command declares `text` to be the title. Use `\\` to tell LaTeX where to start a new line in a long title.

A.15.6 \pagenumbering

`\pagenumbering{num_style}`

Specifies the style of page numbers. Possible values of `num_style` are:

- `arabic` - Arabic numerals
- `roman` - Lowercase Roman numerals
- `Roman` - Uppercase Roman numerals
- `alph` - Lowercase letters
- `Alph` - Uppercase letters

A.15.7 \pagestyle

```
\pagestyle{option}
```

The `\pagestyle` command changes the style from the current page on throughout the remainder of your document.

The valid options are:

- `plain` - Just a plain page number.
- `empty` - Produces empty heads and feet - no page numbers.
- `headings` - Puts running headings on each page. The document style specifies what goes in the headings.
- `myheadings` - You specify what is to go in the heading with the `\markboth` or the `\markright` commands.

A.15.8 \markboth

```
    \markboth{left head}{right head}
```

The `\markboth` command is used in conjunction with the page style `myheadings` for setting both the left and the right heading. You should note that a “left-hand heading” is generated by the last `\markboth` command before the end of the page, while a “right-hand heading” is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

A.15.9 \markright

```
    \markright{right head}
```

The `\markright` command is used in conjunction with the page style `myheadings` for setting the right heading, leaving the left heading unchanged. You should note that a “left-hand heading” is generated by the last `\markboth` command before the end of the page, while a “right-hand heading” is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

A.15.10 \thispagestyle

```
\thispagestyle{option}
```

The `\thispagestyle` command works in the same manner as the `\pagestyle` command except that it changes the style for the current page only.

A.16 Sectioning

Sectioning commands provide the means to structure your text into units.

- `\part`
- `\chapter` (report and book class only)
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

All sectioning commands take the same general form, i.e.,

`\chapter[optional]{title}`

In addition to providing the heading in the text, the mandatory argument of the sectioning command can appear in two other places:

1. The table of contents
2. The running head at the top of the page

You may not want the same thing to appear in these other two places as appears in the text heading. To handle this situation, the sectioning commands have an `optional` argument that provides the text for these other two purposes.

All sectioning commands have `*`-forms that print a *title*, but do not include a number and do not make an entry in the table of contents.

`\appendix`

The `\appendix` command changes the way sectional units are numbered. The `\appendix` command generates no text and does not affect the numbering of parts. The normal use of this command is something like

```
\chapter{The First Chapter}
...
\appendix
\chapter{The First Appendix}
```

A.17 Spaces & Boxes

All the predefined length parameters (see Section A.8.7 [Predefined lengths], page 57) can be used in the arguments of the box-making commands.

A.17.1 \dotfill

The `\dotfill` command produces a “rubber length” that produces dots instead of just spaces.

A.17.2 \hfill

The `\hfill` fill command produces a “rubber length” which can stretch or shrink horizontally. It will be filled with spaces.

A.17.3 \hrulefill

The `\hrulefill` fill command produces a “rubber length” which can stretch or shrink horizontally. It will be filled with a horizontal rule.

A.17.4 \hspace

```
\hspace[*]{length}
```

The `\hspace` command adds horizontal space. The length of the space can be expressed in any terms that LaTeX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\hspace` command. Adding negative space is like backspacing.

LaTeX removes horizontal space that comes at the end of a line. If you don’t want LaTeX to remove this space, include the optional `*` argument. Then the space is never removed.

A.17.5 \addvspace

```
\addvspace{length}
```

The `\addvspace` command normally adds a vertical space of height length. However, if vertical space has already been added to the same point in the output by a previous `\addvspace` command, then this command will not add more space than needed to make the natural length of the total vertical space equal to `length`.

A.17.6 \bigskip

The `\bigskip` command is equivalent to `\vspace{bigskipamount}` where `bigskipamount` is determined by the document class.

A.17.7 \medskip

The `\medskip` command is equivalent to `\vspace{medskipamount}` where `medskipamount` is determined by the document class.

A.17.8 \smallskip

```
\smallskip
```

The `\smallskip` command is equivalent to `\vspace{smallskipamount}` where `smallskipamount` is determined by the document class.

A.17.9 \vfill

The `\vfill` fill command produces a rubber length which can stretch or shrink vertically.

A.17.10 \vspace

`\vspace[*]{length}`

The `\vspace` command adds vertical space. The length of the space can be expressed in any terms that LaTeX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\vspace` command.

LaTeX removes vertical space that comes at the end of a page. If you don't want LaTeX to remove this space, include the optional `*` argument. Then the space is never removed.

A.17.11 \fbox

`\fbox{text}`

The `\fbox` command is exactly the same as the `\mbox` command, except that it puts a frame around the outside of the box that it creates.

A.17.12 \framebox

`\framebox[width][position]{text}`

The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The framebox command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

A.17.13 lrbox

`\begin{lrbox}{cmd} text \end{lrbox}`

This is the environment form of `\sbox`.

The text inside the environment is saved in the box `cmd`, which must have been declared with `\newsavebox`.

A.17.14 \makebox

`\makebox[width][position]{text}`

The `\makebox` command creates a box just wide enough to contain the `text` specified. The width of the box is specified by the optional `width` argument. The position of the text within the box is determined by the optional `position` argument.

- `c` — centred (default)
- `l` — flushleft
- `r` — flushright
- `s` — stretch from left to right margin. The text must contain stretchable space for this to work.

A.17.15 \mbox

`\mbox{text}`

The `\mbox` command creates a box just wide enough to hold the text created by its argument.

Use this command to prevent text from being split across lines.

A.17.16 \newsavebox

`\newsavebox{cmd}`

Declares `cmd`, which must be a command name that is not already defined, to be a bin for saving boxes.

A.17.17 \parbox

`\parbox[position][height][inner-pos]{width}{text}`

A `parbox` is a box whose contents are created in `paragraph` mode. The `\parbox` has two mandatory arguments:

- `width` - specifies the width of the parbox, and
- `text` - the text that goes inside the parbox.

LaTeX will position a `parbox` so its centre lines up with the centre of the text line. The optional *position* argument allows you to line up either the top or bottom line in the parbox (default is top).

If the *height* argument is not given, the box will have the natural height of the text.

The *inner-pos* argument controls the placement of the text inside the box. If it is not specified, *position* is used.

- `t` — text is placed at the top of the box.
- `c` — text is centred in the box.
- `b` — text is placed at the bottom of the box.
- `s` — stretch vertically. The text must contain vertically stretchable space for this to work.

A `\parbox` command is used for a parbox containing a small piece of text, with nothing fancy inside. In particular, you shouldn't use any of the paragraph-making environments inside a `\parbox` argument. For larger pieces of text, including ones containing a paragraph-making environment, you should use a `minipage` environment See Section A.6.13 [minipage], page 46.

A.17.18 \raisebox

`\raisebox{distance}[extend-above][extend-below]{text}`

The `\raisebox` command is used to raise or lower text. The first mandatory argument specifies how high the text is to be raised (or lowered if it is a negative amount). The text itself is processed in `LR mode`.

Sometimes it's useful to make LaTeX think something has a different size than it really does - or a different size than LaTeX would normally think it has. The `\raisebox` command lets you tell LaTeX how tall it is.

The first optional argument, `extend-above`, makes LaTeX think that the text extends above the line by the amount specified. The second optional argument, `extend-below`, makes LaTeX think that the text extends below the line by the amount specified.

A.17.19 \rule

`\rule[raise-height]{width}{thickness}`

The `\rule` command is used to produce horizontal lines. The arguments are defined as follows:

- `raise-height` - specifies how high to raise the rule (optional)
- `width` - specifies the length of the rule (mandatory)
- `thickness` - specifies the thickness of the rule (mandatory)

A.17.20 \savebox

`\savebox{cmd}[width][pos]{text}`

This command typeset `text` in a box just as for `\makebox`. However, instead of printing the resulting box, it saves it in bin `cmd`, which must have been declared with `\newsavebox`.

A.17.21 \sbox

`\sbox{text}`

This commands typeset `text` in a box just as for `\mbox`. However, instead of printing the resulting box, it saves it in bin `cmd`, which must have been declared with `\newsavebox`.

A.17.22 \usebox

`\usebox{cmd}`

Prints the box most recently saved in bin `cmd` by a `\savebox` command.

A.18 Special Characters

The following characters play a special role in LaTeX and are called "special printing characters", or simply "special characters".

`# $ % & ~ _ ^ \ { }`

Whenever you put one of these special characters into your file, you are doing something special. If you simply want the character to be printed just as any other letter, include a `\` in front of the character. For example, `\$` will produce `$` in your output.

One exception to this rule is the `\` itself because `\\` has its own special meaning. A `\` is produced by typing `$\backslash$` in your file.

Also, `\~` means 'place a tilde accent over the following letter', so you will probably want to use `\verb` instead.

In addition, you can access any character of a font once you know its number by using the `\symbol` command. For example, the character used for displaying spaces in the `\verb*` command has the code decimal 32, so it can be typed as `\symbol{32}`.

You can also specify octal numbers with `'` or hexadecimal numbers with `"`, so the previous example could also be written as `\symbol{'40}` or `\symbol{"20}`.

A.19 Splitting the Input

A large document requires a lot of input. Rather than putting the whole input in a single large file, it's more efficient to split it into several smaller ones. Regardless of how many separate files you use, there is one that is the root file; it is the one whose name you type when you run LaTeX.

A.19.1 \include

`\include{file}`

The `\include` command is used in conjunction with the `\includeonly` command for selective inclusion of files. The `file` argument is the first name of a file, denoting '`file.tex`'. If `file` is one the file names in the file list of the `\includeonly` command or if there is no `\includeonly` command, the `\include` command is equivalent to

```
\clearpage \input{file} \clearpage
```

except that if the file '`file.tex`' does not exist, then a warning message rather than an error is produced. If the file is not in the file list, the `\include` command is equivalent to `\clearpage`.

The `\include` command may not appear in the preamble or in a file read by another `\include` command.

A.19.2 \includeonly

`\includeonly{`*file_list*`}`

The `\includeonly` command controls which files will be read in by an `\include` command. *file_list* should be a comma-separated list of filenames. Each filename must match exactly a filename specified in a `\include` command. This command can only appear in the preamble.

A.19.3 \input

`\input{file}`

The `\input` command causes the indicated `file` to be read and processed, exactly as if its contents had been inserted in the current file at that point. The file name may be a complete file name with extension or just a first name, in which case the file '`file.tex`' is used.

A.20 Starting & Ending

Your input file must contain the following commands as a minimum:

```
\documentclass{class}
\begin{document}
  ... your text goes here ...
\end{document}
```

where the `class` selected is one of the valid classes for LaTeX. See Section A.4 [Document Classes], page 40, (and see Chapter 1 [Overview], page 1) for details of the various document classes available locally.

You may include other LaTeX commands between the `\documentclass` and the `\begin{document}` commands (i.e., in the ‘preamble’).

A.21 Table of Contents

A table of contents is produced with the `\tableofcontents` command. You put the command right where you want the table of contents to go; LaTeX does the rest for you. It produces a heading, but it does not automatically start a new page. If you want a new page after the table of contents, include a `\newpage` command after the `\tableofcontents` command.

There are similar commands `\listoffigures` and `\listoftables` for producing a list of figures and a list of tables, respectively. Everything works exactly the same as for the table of contents.

NOTE: If you want any of these items to be generated, you cannot have the `\nofiles` command in your document.

A.21.1 \addcontentsline

`\addcontentsline{file}{sec_unit}{entry}`

The `\addcontentsline` command adds an entry to the specified list or table where:

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `sec_unit` controls the formatting of the entry. It should be one of the following, depending upon the value of the file argument:
    1. `toc` — the name of the sectional unit, such as part or subsection.
    2. `lof` — figure
    3. `lot` — table
- `entry` is the text of the entry.

A.21.2 \addtocontents

`\addtocontents{file}{text}`

The `\addtocontents` command adds text (or formatting commands) directly to the file that generates the table of contents or list of figures or tables.

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `text` is the information to be written.

A.22 Terminal Input/Output

A.22.1 \typein

`\typein[cmd]{msg}`

Prints `msg` on the terminal and causes LaTeX to stop and wait for you to type a line of input, ending with return. If the `cmd` argument is missing, the typed input is processed as if it had been included in the input file in place of the `\typein` command. If the `cmd` argument is present, it must be a command name. This command name is then defined or redefined to be the typed input.

A.22.2 \typeout

`\typeout{msg}`

Prints `msg` on the terminal and in the `log` file. Commands in `msg` that are defined with `\newcommand` or `\renewcommand` are replaced by their definitions before being printed.

LaTeX's usual rules for treating multiple spaces as a single space and ignoring spaces after a command name apply to `msg`. A `\space` command in `msg` causes a single space to be printed. A `^^J` in `msg` prints a newline.

A.23 Typefaces

The `typeface` is specified by giving the "size" and "style". A typeface is also called a "font".

A.23.1 \Styles

The following type style commands are supported by LaTeX.

These commands are used like `\textit{italics text}`. The corresponding command in parenthesis is the "declaration form", which takes no arguments. The scope of the declaration form lasts until the next type style command or the end of the current group.

The declaration forms are cumulative; i.e., you can say `\sffamily\bfseries` to get sans serif boldface.

You can also use the environment form of the declaration forms; e.g. `\begin{ttfamily}...\end{ttfamily}`.

`\textrm (\rmfamily)`
: Roman.

`\textit (\itshape)`
`\emph` Emphasis (toggles between \textit and \textrm).

`\textmd (\mdseries)`
: Medium weight (default). The opposite of boldface.

`\textbf (\bfseries)`
: Boldface.

`\textup (\upshape)`
: Upright (default). The opposite of slanted.

`\textsl (\slshape)`
: Slanted.

`\textsf (\sffamily)`
: Sans serif.

`\textsc (\scshape)`
: Small caps.

`\texttt (\ttfamily)`
: Typewriter.

`\textnormal (\normalfont)`
: Main document font.

`\mathrm` Roman, for use in math mode.

`\mathbf` Boldface, for use in math mode.

`\mathsf` Sans serif, for use in math mode.

`\mathtt` Typewriter, for use in math mode.

`\mathit` Italics, for use in math mode, e.g. variable names with several letters.

`\mathnormal`
: For use in math mode, e.g. inside another type style declaration.

`\mathcal` 'Calligraphic' letters, for use in math mode.

In addition, the command `\mathversion{bold}` can be used for switching to bold letters and symbols in formulas. `\mathversion{normal}` restores the default.

A.23.2 Sizes

The following standard type size commands are supported by LaTeX.

The commands as listed here are "declaration forms". The scope of the declaration form lasts until the next type style command or the end of the current group.

You can also use the environment form of these commands; e.g. `\begin{tiny}...\end{tiny}`.

`\tiny`

`\scriptsize`
`\footnotesize`
`\small`

`\normalsize`
: (default)

`\large`

`\Large`

`\LARGE`

`\huge`

`\Huge`

A.23.3 Low-level font commands

These commands are primarily intended for writers of macros and packages. The commands listed here are only a subset of the available ones. For full details, you should consult Chapter 7 of *The LaTeX Companion*.

`\fontencoding{enc}`
: Select font encoding. Valid encodings include `OT1` and `T1`.

`\fontfamily{family}`
: Select font family. Valid families include:

- `cmr` for Computer Modern Roman
- `cmss` for Computer Modern Sans Serif
- `cmtt` for Computer Modern Typewriter

and numerous others.

`\fontseries{series}`
: Select font series. Valid series include:

- `m` Medium (normal)
- `b` Bold

- `c` Condensed
- `bc` Bold condensed
- `bx` Bold extended

and various other combinations.

`\fontshape{shape}`
: Select font shape. Valid shapes are:
  - `n` Upright (normal)
  - `it` Italic
  - `sl` Slanted (oblique)
  - `sc` Small caps
  - `ui` Upright italics
  - `ol` Outline

  The two last shapes are not available for most font families.

`\fontsize{size}{skip}`
: Set font size. The first parameter is the font size to switch to; the second is the `\baselineskip` to use. The unit of both parameters defaults to pt. A rule of thumb is that the baselineskip should be 1.2 times the font size.

`\selectfont`
: The changes made by calling the four font commands described above do not come into effect until `\selectfont` is called.

`\usefont{enc}{family}{series}{shape}`
: Equivalent to calling `\fontencoding`, `\fontfamily`, `\fontseries` and `\fontshape` with the given parameters, followed by `\selectfont`.

Appendix B Example Documents

B.1 Skeleton Document

This is an example of a skeletal LaTeX document in source form. Comments are included to help explain, but they are of course not needed in an actual document.

```
% Almost nothing except comments can go before the \documentclass
% command, which must occur in every LaTeX 2e document.

\documentclass{article}
% Other common classes are ‘book’, ‘report’, and ‘letter’.

% This area before \begin{document} is called the ‘preamble’.
% It may include various things, such as definitions,
% new lengths or variables, but it may not include anything that
% would cause LaTeX to try to print anything.

\begin{document}

% The body of the document goes here.  This one doesn’t have anything.

\end{document}

% Anything down here will be ignored by LaTeX.
```

B.2 Example Article

This is an example LaTeX article in source form.

```
\documentclass{article}

\title{Foobars:\\Are They Good For You?}
\author{The Foobar Marketing Association}

\begin{document}

\maketitle

\section{What is a Foobar?}

A Foobar is a wonderfully tasty treat that you can apply to any program
of your choice for added flavor and seasoning.  Unlike ordinary bars,
Foobars have such features as:

\begin{itemize}
\item A fooey taste that cannot be denied.
```

```
\item Baz flavoring at no extra charge.
\item Consistency that can't be matched
via our Qux manufacturing process.
\end{itemize}

\section{What Foobars Can Do For You (And Your Country)}

Unlike ordinary bars, that sit around until digested, Foobars can be
used to help debug your programs.  Either Foo, Bar, or Foobar can be
inserted into an otherwise normal looking program to turn it Foolicious!
Peppered with Foos and Bars, you're sure to attract some bugs until
you can stuff them out with a handy tool such as the GNU Debugger (As
Seen on FTP!).

\section{Why You Should Use Foobars Now}

Imagine the embarassement of showing your best buddies some cool new
code you wrote, only to have it blow up in your face.  Didn't use
Foobar, did you?  Had you used a Foobar, that bug may have been found
earlier, preventing potentially years of teasing at your expense.

So don't delay.  Use Foobars today!

\end{document}
```

B.3 Example Report

B.4 Example Book

B.5 Example Letter

This is an example letter in LaTeX source form.

```
\documentclass{letter}

\begin{document}

\begin{letter}%
{Mr. President\\1600 Pennsylvania Ave.\\Washington, D.C.}

\cc{Secretary of Defense\\Secretary of the Treasury}

\name{R. W. Drofnats}

\address{R. W. Droftats%
\\The James T. Kirk Lab of Computer Science\\Orion Nebula}
```

```
\opening

Dear Mr. President,

It has come to my attention that Foobars could be a great aid to
the people of America and those dwelling on planet earth.  Countless
millions could be saved on such trivial tasks as software development
and be used for more productive purposes, such as $\$20,000$ toilet
seats for the Pentagon.

What is a Foobar?  Well, to put it simply, it's the latest wave to hit
the New Economy: it's the Next Big Thing, the Killer App, the next
Pokemon.  It will revolutionize the way we do business, run the
military, and eat our toast.  Its applications are as limitless as the
blades of grass on the South Lawn.

I'm sure you want to know more.  So feel free to contact me any time
via subspace (the Galactic Security Administration should have my
address, shoe size, relationship to Fidel Castro and anything else you
need to know that's vital for Galactic Security).  Let's both work
together and highten our synergy to make Foobar bridge the gaps in the
Galactic Economy.  Let us strive towards greatness, pompous excess,
hubris, and even more excessive hoarding of wealth than ever before!

\closing{Sincerely,}

\signature{R. W. Drofnats\\{\sc nomad} Investigative Section}

\end{letter}

\end{document}
```

B.6 Example Slides

Appendix C Cookbook

C.1 Nifty Tables with `halign`

C.2 Writing a Dictionary for Dummies

C.3 Object Oriented Personas

C.4 A Neat Program Written in LaTeX

C.5 A Literate Program in Perl

C.6 Using Make to Build LaTeX

Appendix D Other Resources

For a description of what goes on inside TEX (a must for any serious TeXnician), you should consult:

The TeXbook by Donald E. Knuth, ISBN 0-201-13448-9, published jointly by the American Mathematical Society and Addison-Wesley Publishing Company.

For a basic to intermediate description of LATEX, you should consult:

LaTeX: A Document Preparation System, by Leslie Lamport, Addison-Wesley Publishing Company, 2nd edition, 1994.

A more detailed description of LATEX (primarily extension packages) can be found in:

The LaTeX Companion, by Michel Goossens, Frank Mittelbach, and Alexander Samarin, Addison-Wesley, 1994.

There is also an excellent group of TEX and LATEX users, known as TUGboat (The TEX User's Group). You can find them at `http://www.tug.org`.

Appendix E Concept Index

A

Accessing any character of a font 76
Adobe Acrobat PDF viewer 9
Aligning Equations 43
Appendicies 23
Appendicies, making a bibliography 23
Appendicies, making an index 23
Appendix 23
Appendix, creating 71
Arrays, math 42
article class 40
Article, example 85
Article, making a 23
Author, for titlepage 69

B

Bibliography, creating 23
Bibliography, creating (automatically) 54
Bibliography, creating (manually) 53
BibTeX, using 23, 54
book class 40
Book, example 86
Book, making a 23
Boxes 72

C

Calculus, typsetting 27
Cc list 58
Centering text 42
Centering text, environment for 42
Chapters 23
Characters, reserved 76
Characters, special 76
Citations 23
Classes of document 40
Classes, document 23
Code, typsetting source 29
Colons, typing 17
Columns, document formatting 31
Commands 33
Commands, creating 33
Commands, declarations 13
Commands, defining new ones 38
Commands, syntax 11
Commas, typing 17
Comments 11
Computer Documentation, typesetting 29
Computer programs, typesetting 55
Computer Programs, typesetting 29
Constants, typesetting computer 29
Cookbook, LaTeX 89
Counters, a list of 35
Counters, creating 36
Counters, getting the value of 36
Counters, setting 36
Creating letters 58
Creating pictures 46
Creating tables 51
Cross referencing 37
Cross referencing using page number 37
Cross referencing using section number 37
Cross-references 23
Customizing documents 31
Customizing fonts 31
Customizing lengths 31
Customizing lists 31

D

Dashes, typing 18
Date, for titlepage 69
Declarations 13
Defining a new command 38
Defining new environments 38
Defining new fonts 39
Defining new theorems 38
Description environment 25
Device Indepentant Format, making 8
Dictionary, writing a 89
Display math 27
Displaying quoted text 49
Displaying quoted text with paragraph indentation 49
DocBook and LaTeX compared 4
Document Classes 40
Document types 23
Documentation, typesetting computer 29
Documents, appendicies 23
Documents, article example 85
Documents, basic structure 85
Documents, bibliography 23
Documents, book example 86
Documents, chapters 23
Documents, citations 23
Documents, classes 23
Documents, double sided 31
Documents, example 85
Documents, example article 85
Documents, example book 86
Documents, example letter 86
Documents, example report 86
Documents, example slides 87
Documents, figures 23
Documents, font selection 31
Documents, indicies 23
Documents, layout 25

Documents, letter example . . . 86
Documents, logical strucure . . . 23
Documents, options . . . 31
Documents, parts . . . 23
Documents, references . . . 23
Documents, report example . . . 86
Documents, sections . . . 23
Documents, single column . . . 31
Documents, single sided . . . 31
Documents, skeleton . . . 85
Documents, slides example . . . 87
Documents, table of contents . . . 23
Documents, tables . . . 23
Documents, two column . . . 31
Double sided documents . . . 31
Drawing images . . . 25
DVI, making from LaTeX . . . 8
`dvips` command . . . 8

E

Enclosed material . . . 59
Ending & Starting . . . 78
Ending sentences . . . 16
Enlarge current page . . . 61
Enumerate environment . . . 25
Environments . . . 42
Environments, defining . . . 38
Environments, syntax . . . 12
EPS, including images . . . 25
Equations, aligning . . . 43
Equations, environment for . . . 44
Equations, typsetting . . . 27
Examples, article . . . 85
Examples, book . . . 86
Examples, complex table . . . 89
Examples, dictionary . . . 89
Examples, document . . . 85
Examples, LaTeX program . . . 89
Examples, letter . . . 86
Examples, literate program . . . 89
Examples, object oriented personas . . . 89
Examples, program in LaTeX . . . 89
Examples, report . . . 86
Examples, slides . . . 87
Exmaples, perl literate program . . . 89
exponent . . . 65

F

Figures . . . 23
Figures, footnotes in . . . 46
Files input to LaTeX . . . 7
Files output from LaTeX . . . 7
Flushing a page . . . 61
Font commands, low-level . . . 82
Font Sizes . . . 82
Font Styles . . . 81
Fonts . . . 19, 81
Fonts, customizing . . . 31
Fonts, new commands for . . . 39
Footnotes in figures . . . 46
Footnotes, creating . . . 56
Formatting Text . . . 42
Formulae, environment for . . . 44
Formulae, maths . . . 65
Fractions, typsetting . . . 27
French, typesetting . . . 22
Functions, declaring computer . . . 29

G

German, typesetting . . . 22
Getting LaTeX to run . . . 7
Global options . . . 40
Glossary, making . . . 25
Greek letters . . . 65

H

halign, using . . . 89
Horizontal spacing . . . 25
How to run LaTeX . . . 7
How to start using LaTeX . . . 7
HTML and LaTeX compared . . . 4
HTML can be generated from LaTeX . . . 4
HTML, making with LaTeX . . . 9
Hyphenation, defining . . . 62
Hyphenation, forcing . . . 61
Hyphenization . . . 18
Hyphens, typing . . . 18
Hyphens, using . . . 18

I

Images, drawing . . . 25
Images, EPS . . . 25
Images, including . . . 25
Images, PostScript . . . 25
Including images . . . 25
Indent, forcing . . . 64
Indent, suppressing . . . 64
Index . . . 23
Indicies, creating . . . 23
Indicies, files . . . 8
Indicies, generating . . . 8
Input file, splitting . . . 76
Input files for LaTeX . . . 7
Input spacing . . . 15
Input/Output . . . 80
Inserting figures . . . 44
Internals . . . 33
Internals, LaTeX . . . 33
Internals, LaTeX programs . . . 33
Introduction to LaTeX . . . 1
Invoking LaTeX . . . 7

Item separation, customizing . . . 31
Itemize environment . . . 25

J

Jump start to using LaTeX . . . 7
Justification, ragged left . . . 45
Justification, ragged right . . . 44

L

Labelled lists, creating . . . 43
Labels . . . 23
Languages, typesetting foreign . . . 20
LaTeX and plain text . . . 15
LaTeX can be converted to HTML . . . 4
`latex` command . . . 8
LaTeX Cookbook . . . 89
LaTeX document processing . . . 7
LaTeX for dummies . . . 7
LaTeX input files . . . 7
LaTeX internals . . . 33
LaTeX introduction . . . 1
LaTeX modes . . . 11
LaTeX output files . . . 7
LaTeX overview . . . 1
LaTeX programs . . . 33
LaTeX syntax . . . 11
LaTeX to DVI . . . 8
LaTeX to HTML . . . 9
LaTeX to PostScript . . . 8
LaTeX versus DocBook . . . 4
LaTeX versus HTML . . . 4
LaTeX versus TEX . . . 3
LaTeX versus WYSIWYG . . . 4
LaTeX versus XML . . . 4
LaTeX, building with make . . . 89
latex2html . . . 9
Layout . . . 25
Layout commands . . . 40
Layout, lists . . . 25
Layout, page . . . 25
Left-justifying text . . . 44
Left-justifying text, environment for . . . 44
Left-to-right mode . . . 68
Lengths . . . 31
Lengths, adding to . . . 57
Lengths, defining a new . . . 57
Lengths, defining and using . . . 57
Lengths, predefined . . . 57
Lengths, setting value of . . . 57
letter class . . . 40
Letter, example . . . 86
Letter, making a . . . 23
Letters . . . 58
Letters, ending . . . 58
Letters, starting . . . 59
Line Breaking . . . 61
Line breaks . . . 62
Lines in tables . . . 51
Lining text up in columns using tab stops . . . 50
Lining text up in tables . . . 51
Lists of items . . . 45
Lists of items, generic . . . 46
Lists of items, numbered . . . 43
Lists, customizing . . . 31
Lists, layout . . . 25
Literal text, verbatim environment . . . 25
Literate programming . . . 29
Loading additional packages . . . 40
Logical structure . . . 23
Low-level font commands . . . 82
LR mode . . . 14, 68
LyX (reference to) . . . 4

M

Macros, creating . . . 33
Make, using to build LaTeX . . . 89
makeindex . . . 23
Making a title page . . . 55
Making paragraphs . . . 64
Making PDF documents . . . 9
Margin Notes . . . 65
Math Formulae . . . 65
Math mode . . . 14, 68
Math mode, display enviornment . . . 27
Math mode, entering . . . 65
Math mode, spacing . . . 66
Math Mode, typesetting for . . . 27
Mathematics, basic . . . 27
Mathematics, elaborate typsetting . . . 27
Mathematics, equations . . . 27
Mathematics, indexing automatically . . . 27
Mathematics, large sections . . . 27
Mathematics, theorems . . . 27
Mathematics, typesetting . . . 27
Maths Miscellany . . . 66
Maths symbols . . . 65
Matricies, typsetting . . . 27
Measures . . . 31
MetaFont . . . 31
Minipage, creating a . . . 46
Modes . . . 14, 68
Modes, LaTeX . . . 11
Modes, LR . . . 14
Modes, math . . . 14
Modes, paragraph . . . 14
Modules, defining computer . . . 29
MSWord (LaTeX contrasted with) . . . 4
Multicolumn text . . . 41

N

New line, starting . . . 61
New line, starting (paragraph mode) . . . 62

New Page 62
NFSS 31
Notes in the margin 65

O

Object oriented personas 89
Options for documents 31
Options, global 40
Outline, making 25
Output files for LaTeX 7
Overview of LaTeX 1

P

Packages 33
Packages, custom 33
Packages, loading 33, 40
Packages, using 33
Packages, writing 33
Page break, forcing 62
Page Breaking 61
Page Formatting 61
Page layout 25
Page numbering 69
Page styles 69
Paragraph mode 14, 68
Paragraph, starting a new 64
Paragraphs 64
Parts 23
PDF viewers 9
PDF, making from LaTeX 9
pdflatex, using 9
pdftex, relation of pdflatex to 9
Periods, spacing after 16
Perl, literate programming in 89
Personas, object oriented 89
Pictures, creating 46
Plain text 15
Plain text, foreign languages 20
Plain text, hyphens 18
Plain text, special characters 15
Plain text, typefaces 19
Plain text, typing punctuation 17
Poetry, an environment for 55
Poetry, verse environment 25
Portable Document Format, making 9
PostScript, including images 25
PostScript, making from LaTeX 8
Predefined lengths 57
Pretty-printing, source code 29
Processing LaTeX documents 7
Programming LaTeX 33
Programming, literate 29
Programs, LaTeX example 89
Programs, literate example 89
Programs, perl example 89
Programs, typesetting 55
Programs, typsetting 29
Punctuation, how to type 17
Punctuation, spacing 16

Q

Quotation enviornment 25
Quotation marks, typing 17
Quotations, environments 25
Quote environment 25
Quoted text with paragraph indentation, displaying 49
Quoted text, displaying 49

R

Ragged left text 45
Ragged left text, environment for 45
Ragged right text 44
Ragged right text, environment for 44
Reasons for using LaTeX 1
Reasons not to use LaTeX 3
References 23
Remarks in the margin 65
report class 40
Report, example 86
Report, making a 23
Reserved Characters 76
Right-justifying text 45
Right-justifying text, environment for 45
Running LaTeX 7

S

Scope 13
Sectioning 71
Sections 23
Semicolons, typing 17
Simulating typed text 55
Single column documents 31
Single sided documents 31
Sizes of text 82
Slides, example 87
Slides, making 23
Source Code, basic typesetting 29
Space, inserting vertical 72
Spaces 72
Spacing on page 25
Spacing, input file 15
Spacing, output 16
Spacing, within Math mode 66
Spanish, typesetting 22
Special Characters 76
Special Characters, caveats 15
Special Characters, typing 15
Splitting the input file 76
StarOffice (LaTeX contrasted with) 4
Start using LaTeX 7

Starting & Ending 78
Starting on a right-hand page 61
Structure, chapters 23
Structure, documents 23
Structure, parts 23
Structure, sections 23
Styles of text 81
Styles, page 69
Subscript 65
Superscript 65
Symbols 65
Syntax, basic 11
Syntax, commands 11
Syntax, environments 12
Syntax, LaTeX 11

T

Tab stops, using 50
Table of Contents, creating 79
Table of contents, making 23
Tables, creating 51
Tables, making with tabular 25
Tables, using hailgn 89
Tabular 23
Tabular environment 25
Terminal Input/Output 80
TEX and LaTeX compared 3
Text, how to type in LaTeX 15
Thanks, for titlepage 69
Theorems, defining 38
Theorems, typesetting 54
Theorems, typsetting 27
Title making 69
Title pages, creating 55
Title, for titlepage 69
Two column documents 31
Typed text, simulating 55
Typeface Sizes 82
Typeface Styles 81
Typefaces 81
Types of documents 23
Typesetting plain text 15

U

Using BibTeX 54
Using LaTeX 7

V

Variables, a list of 35
Variables, typesetting computer 29
Verbatim environment 25
Verbatim text 55
Verbatim, using for source code 29
Verse environment 25
Vertical space, inserting 72
Vertical spacing 25

W

Why you should not use LaTeX 3
Why you should use LaTeX 1
WordPerfect (LaTeX contrasted with) 4
WYSIWYG formatting and LaTeX compared . . . 4

X

XML and LaTeX compared 4
xpdf PDF viewer 9

Appendix F Command Index

$

$ 65

^

^ 65

_

_ 65

\

\! 66
\' (tabbing) 50
\(.... 65
\) 65
\+ 50
\, ... 66
\- (hyphenation) 61
\- (tabbing) 50
\; 66
\< 50
\= 50
\> 50
\[.... 65
\] 65
\` (tabbing) 50
\\ 61
\a 51
\addcontentsline 79
\address 58
\addtocontents 79
\addtocounter 35
\addtolength 57
\addvspace 72
\alph 35
\Alph 35
\appendix 71
\arabic 35
\author 69
\backslash 76
\begin 42
\bfseries 81
\bibitem 53
\bibliography 54
\bibliographystyle 54
\bigskip 72
\caption 44
\cc 58
\cdots 66
\centering 42
\chapter 71
\circle 47
\cite 54
\cleardoublepage 61
\clearpage 61
\cline 53
\closing 58
\COLON 66
\dashbox 47
\date 69
\ddots 66
\depth 57
\documentclass 40
\dotfill 72
\emph 81
\encl 59
\end 42
\enlargethispage 61
\fbox 73
\flushbottom 41
\fnsymbol 35
\fontencoding 82
\fontfamily 82
\fontseries 82
\fontshape 83
\fontsize 83
\footnote 56
\footnotemark 56
\footnotesize 82
\footnotetext 56
\frac 66
\frame 48
\framebox 48, 73
\fussy 61
\height 57
\hfill 72
\hline 53
\hrulefill 72
\hspace 72
\huge 82
\Huge 82
\hyphenation 62
\include 76
\includeonly 76
\indent 64
\input 77
\item 45
\itshape 81
\kill 50
\label 37
\large 82
\Large 82
\LARGE 82
\ldots 66
\lefteqn 43
\line 48

\linebreak . . . 62
\linethickness . . . 48
\listoffigures . . . 79
\listoftables . . . 79
\location . . . 59
\makebox . . . 73
\makebox (picture) . . . 48
\makelabels . . . 59
\maketitle . . . 69
\markboth . . . 70
\markright . . . 70
\mathbf . . . 81
\mathcal . . . 82
\mathit . . . 81
\mathnormal . . . 82
\mathrm . . . 81
\mathsf . . . 81
\mathtt . . . 81
\mathversion . . . 82
\mbox . . . 73
\mdseries . . . 81
\medskip . . . 72
\multicolumn . . . 53
\multiput . . . 48
\name . . . 59
\newcommand . . . 38
\newcounter . . . 36
\newenvironment . . . 38
\newfont . . . 39
\newlength . . . 57
\newline . . . 62
\newpage . . . 62
\newsavebox . . . 74
\newtheorem . . . 38
\nocite . . . 54
\nofiles . . . 79
\noindent . . . 64
\nolinebreak . . . 62
\nopagebreak . . . 62
\normalfont . . . 81
\normalsize . . . 82
\onecolumn . . . 41
\opening . . . 59
\oval . . . 49
\overbrace . . . 66
\overline . . . 66
\pagebreak . . . 62
\pagenumbering . . . 69
\pageref . . . 37
\pagestyle . . . 70
\par . . . 64
\paragraph . . . 71
\parbox . . . 74
\ps . . . 59
\pushtabs . . . 51
\put . . . 49
\raggedbottom . . . 41
\raggedleft . . . 45
\raggedright . . . 44
\raisebox . . . 74
\ref . . . 37
\refstepcounter . . . 36
\rmfamily . . . 81
\roman . . . 36
\Roman . . . 36
\rule . . . 75
\savebox . . . 75
\sbox . . . 75
\scriptsize . . . 82
\scshape . . . 81
\selectfont . . . 83
\setcounter . . . 36
\setlength . . . 57
\settodepth . . . 57
\settoheight . . . 57
\settowidth . . . 57
\sffamily . . . 81
\shortstack . . . 49
\signature . . . 59
\sloppy . . . 63
\slshape . . . 81
\small . . . 82
\smallskip . . . 72
\sqrt . . . 66
\startbreaks . . . 59
\stepcounter . . . 36
\stopbreaks . . . 59
\subparagraph . . . 71
\subsection . . . 71
\subsubsection . . . 71
\symbol . . . 76
\tableofcontents . . . 79
\telephone . . . 60
\textbf . . . 81
\textit . . . 81
\textmd . . . 81
\textnormal . . . 81
\textrm . . . 81
\textsc . . . 81
\textsf . . . 81
\textsl . . . 81
\texttt . . . 81
\textup . . . 81
\thanks . . . 69
\thispagestyle . . . 70
\tiny . . . 82
\title . . . 69
\today . . . 55
\totalheight . . . 57
\ttfamily . . . 81
\twocolumn . . . 41
\typein . . . 80
\typeout . . . 80
\underbrace . . . 66
\underline . . . 66
\upshape . . . 81

`\usebox` 75
`\usecounter` 36
`\usefont` 83
`\usepackage` 40
`\value` 36
`\vdots` 67
`\vector` 49
`\verb` 55
`\vfill` 73
`\vline` 53
`\vspace` 73
`\width` 57

A

`array` 42

C

`center` 42

D

`description` 43

E

`enumerate` 43
`eqnarray` 43
`equation` 44

F

`figure` 44
`flushleft` 44
`flushright` 45

I

`itemize` 45

L

`letter` 45
`list` 46
`lrbox` 73

M

`minipage` 46

P

`picture` 46

Q

`quotation` 49
`quote` 49

T

`tabbing` 50
`table` 51
`tabular` 51
`thebibliography` 53
`theorem` 54
`titlepage` 55

V

`verbatim` 55
`verse` 55